UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería informática

TRABAJO FIN DE GRADO

Una aplicación para el entrenamiento a intervalos para smartwatch

González Vázquez, Jorge
Tutor: Gómez Escribano, Javier
Ponente (si procede): Montoro Manrique, Germán

Fecha: Mayo, 2018

Una aplicación para el entrenamiento a intervalos para smartwatch

AUTOR: Jorge González Vázquez
TUTOR: Javier Gómez Escribano

Grupo de la EPS (opcional)
Dpto. GII
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2018

Resumen (castellano)

En la actualidad, cada vez es más común que las personas utilicen smartwatch en su día a día, como complemento del teléfono móvil para recibir notificaciones de las diferentes aplicaciones o utilizar en el mismo smartwatch aplicaciones.

Muchas de las aplicaciones de smartwatch están relacionadas con el deporte, ya que para muchos usuarios es más cómodo salir a entrenar con un dispositivo como un smartwatch que con un Smartphone. Esta es una de las principales razones por las que muchos de los usuarios utilizan estos dispositivos principalmente para entrenar, y como consecuencia se ha abierto un mercado de relojes deportivos muy amplio y de aplicaciones que intentan satisfacer las necesidades de los usuarios a la hora de entrenar.

En este trabajo de fin de grado se desarrolla una aplicación para smartwatch en Android Wear, para realizar entrenamientos de diferentes deportes, centrándose en el entrenamiento por intervalos de diferentes tipos. Este tipo de entrenamiento es muy normal cuando se llega a cierto nivel, y hay muy pocas aplicaciones en las que se gestionen de manera automática y se puedan configurar directamente en el smartwatch sin necesidad de utilizar una aplicación web o móvil. También se busca que el usuario pueda configurar a su gusto los entrenamientos, pudiendo programar avisos, para que se le notifique información durante el entrenamiento y pudiendo elegir la información que quiere que se muestre en la pantalla del smartwatch mientras realiza el entrenamiento.

Además, la aplicación almacenará los entrenamientos que realice el usuario y sus estadísticas, para que pueda consultar el historial de entrenamientos que realiza, y llevar un seguimiento de sus progresos.

Para realizar este proyecto una de las partes fundamentales, es el uso de la API de *Google Fit*, con la que se ha desarrollado las principales funcionalidades de la APP. Antes de ello se ha buscado y analizado el funcionamiento de esta API para usarla correctamente.

Se puede decir que el resultado del proyecto ha sido satisfactorio, con la creación de una aplicación que satisface lo que estábamos buscando.

Palabras clave (castellano)

Smartwatch, Android Wear, intervalos, Google fit, sensores, entrenamiento

Abstract

Currently, it is increasingly common for people to use smartwatch in their day to day, as a complement to the mobile phone to receive notifications of different applications or use in the same smartwatch applications.

Many of the smartwatch applications are related to sports, since for many users it is more convenient to go out and train with a device such as a smartwatch than with a Smartphone. This is one of the main reasons why many of the users use these devices mainly for training, and as a consequence a wide market of sports watches has been opened and of applications that try to satisfy the needs of the users when it comes to training.

In this end-of-grade project, an application for smartwatch is developed on Android Wear, to train different sports, focusing on training at different intervals. This type of training is very normal when it reaches a certain level, and there are very few applications in which they are managed automatically and can be configured directly on the smartwatch without the need to use a web or mobile application. It is also intended that the user can configure training to your liking, being able to schedule notices, so that you are notified of information during training and can choose the information you want to be displayed on the screen of the smartwatch while doing the training.

In addition, the application will store the trainings carried out by the user and his statistics, so that he can consult the training history he performs, and keep track of his progress.

To carry out this project one of the fundamental parts is the use of the Google Fit API, with which the main functionalities of the APP have been developed. Before this, the functioning of this API has been searched and analyzed to use it correctly.

It can be said that the result of the project has been satisfactory, with the creation of an application that satisfies what we were looking for.

Keywords

Smartwatch, Android Wear, intervals, Google fit, sensors, training

Agradecimientos

En este punto me gustaría agradecer a todas las personas que me han ayudado o me han animado hasta lograr finalizar mis estudios y en consecuencia este último paso al realizar este trabajo de fin de grado.

En primer lugar, me gustaría agradecérselo a todos los profesores que he ido teniendo a lo largo de la carrera, y en especial a mi tutor del TFG, Javier Gómez Escribano, por permitirme realizar este proyecto y por toda la ayuda que me has ofrecido.

A mis amigos y a mi novia, gracias por animarme cuando estaba agobiado con las prácticas o los exámenes.

Finalmente, agradecer a mi familia, que siempre me ha apoyado en todo y me ha ayudado todo lo posible para que finalice mis estudios.

¡GRACIAS!

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

La práctica de deportes para cuidar la salud y estar en forma es cada vez más común en nuestra sociedad. Esto da paso a mucha gente que cada vez se toma más en serio sus entrenamientos semanales, principalmente el running, y quieren monitorizar cada uno de ellos. Para ello el uso de smartwatchs deportivos o pulseras de actividad es una realidad. Cada vez está más al orden del día la gente que entrena con uno de estos aparatos, y nosotros queremos desarrollar una aplicación que se ajuste lo más posible a las necesidades de los deportistas.

Esta situación mezcla deporte con tecnología. La aplicación facilitará al usuario los parámetros más importantes del entrenamiento que realice y le permitirá crear un entrenamiento personalizado en todos los aspectos. La aplicación se va a centrar en tres deportes en cuestión que ahora están muy de moda, como son el running, el ciclismo y la natación, y en el entrenamiento por intervalos. Todo ello será realizado directamente en el smartwatch, sin necesidad de una aplicación móvil ni una aplicación web.

Para poder monitorizar la actividad del usuario vamos a utilizar la API de *Google Fit*, la cual ha sido muy útil para el desarrollo de la aplicación y se explica en profundidad en el anexo C.

1.1 Motivación y objetivos

La principal motivación que tenemos es conseguir realizar una aplicación para smartwatch que cumpla con los requisitos que los usuarios demandan, encontrando un hueco en el mercado que creemos que todavía está por ocupar, ya que como veremos creemos que todavía se pueden mejorar las aplicaciones que hay en la actualidad.

Una de las funcionalidades que se ha echado en falta en muchas de las aplicaciones es el entrenamiento por intervalos, y en caso de contar con ella en muchos casos es muy precaria o tiene un uso un poco complejo para el usuario. Por ello queremos que nuestra aplicación cuente con una sección de entrenamiento por intervalos muy intuitiva y útil para el usuario.

La aplicación que queremos desarrollar tendría unos objetivos claros que son:

- Conocer qué funcionalidades son más importantes para el usuario.
- Crear una aplicación con una interfaz agradable para el usuario.
- Que toda la funcionalidad de la aplicación se realice en el smartwatch, sin necesidad de hacer uso de un móvil o una aplicación web.
- Que el usuario pueda crear entrenamientos personalizados a su gusto, incluyendo entrenamientos por intervalos con la mayor configuración posible.
- Monitorizar la actividad física del usuario y que los datos registrados sean precisos y correctos.

1.2 Organización de la memoria

La memoria consta de los siguientes capítulos:

Estado del arte: En esta sección se habla de los smartwatchs deportivos que ofrece el mercado y de las aplicaciones más populares para hacer deporte. Se realiza un análisis de las funcionalidades que proponemos y las que ya tienen las aplicaciones.

Diseño: En esta sección se realiza un diseño de la aplicación, teniendo en cuenta un formulario que se ha realizado a una serie de usuarios. Después se enumeran los requisitos funcionales y no funcionales con los que cuenta la aplicación y se analiza los actores que interactúan con el sistema y los casos de uso que ellos realizan. Finalmente, se comprueba si el análisis de requisitos y los casos de uso están bien realizados mediante una matriz de trazabilidad.

Desarrollo: En esta sección se explica una serie de partes de la implementación de la aplicación que se han considerado importantes, adjuntando capturas de algunas partes del código. Principalmente se explica cómo se ha interactuado con la API de *Google Fit* en la aplicación.

Integración y pruebas: En esta sección se enumeran las pruebas que se han realizado para comprobar el correcto funcionamiento de la aplicación.

2 Estado del arte

2.1 Relojes deportivos

En la actualidad el mercado de relojes deportivos es muy amplio. Es un campo que no para de crecer, dado a la necesidad de los usuarios de monitorizar su actividad física, para así poder llevar un control de sus entrenamientos, como hemos recalcado anteriormente.

En las ilustraciones 0-1 y 0-2 del anexo A podemos ver algunos de los relojes deportivos más demandados en la actualidad. Nos hemos centrado en los que cuentan con GPS y pulsómetro como es el caso del reloj con el que se ha trabajado para desarrollar la aplicación (*Huawei watch 2*). Al realizar este estudio, podemos decir que hay dos marcas por encima del resto, que son: Garmin y Polar. Estas marcas son referentes como relojes deportivos. Por ello uno de nuestros objetivos es acercarnos al rendimiento de estos relojes con una aplicación para Android Wear.

Garmin y Polar, tienen integrada su propia aplicación para entrenar. Las cuales son las más completas del mercado con diferencia. El problema de estos relojes es que en general solo sirven para entrenar, no tienen soporte para otras aplicaciones, aunque están trabajando en ello. De hecho, el modelo Polar recogido en la ilustración 0-2 del anexo A es compatible con Android Wear.

Realizando un análisis de lo que nos ofrecen estos relojes a la hora de realizar entrenamientos por intervalos, nos damos cuenta de que, en el caso de Polar, este tipo de entrenamientos tienen que ser configurados desde su aplicación móvil o web (*Polar Flow*), y en el caso de Garmin sí que cuenta con la posibilidad de configurar un entrenamiento por intervalos bastante completo desde el smartwatch, pero sin llegar a serlo tanto como Polar. Por lo tanto, nosotros queremos hacer una mezcla de ambos. Poder realizar tanta configuración del entrenamiento como Polar y que este proceso se realice de manera intuitiva desde el Smartwatch sin necesidad de otro sistema como Garmin. [1] [2]

2.2 Aplicaciones relacionadas

Además del estudio de los relojes y de las funcionalidades de algunos de ellos, se ha realizado un estudio de algunas de las aplicaciones más populares de entrenamiento:

- **Strava**: Aplicación para móvil o smartwatch. Ideal para realizar deporte al aire libre. Monitoriza la actividad y la registra. Forma una comunidad en la que los miembros comparten sus entrenamientos y resultados. Tiene versión Premium que añade algunas funcionalidades. [3]
- **Runstatic**: Aplicación para móvil o smartwatch. Monitoriza y registra tu actividad física. Puedes ponerte metas y te registra los logros que vas consiguiendo. Te da la posibilidad de conectarse con *Garmin Connect* o de integrarse con *Google Fit*. Cuenta con un reproductor de música integrado.[4]
- **Runkeeper**: Aplicación para móvil o smartwatch. Monitoriza y registra tu actividad física. Mide la evolución de tu rendimiento, almacenando un historial detallado de tus actividades y mide tus progresos con logros y metas. Permite compartir tus entrenamientos mediante Facebook o Twitter. [5]

- **Nike plus**: Aplicación de running para móvil, no tiene versión para smartwatch. Hace un seguimiento de tus carreras y guarda los registros. Permite personalizar entrenamientos con un horario determinado. Te permite compartir los entrenamientos con tus amigos. [6]
- **Google Fit**: Registra la actividad que realice el usuario. Puedes fijarte objetivos y la aplicación te avisará cuando estos sean cumplidos. Mide tu bienestar, recopilando información de otras aplicaciones deportivas o de nutrición. Aplicación tanto para móvil como para smartwatch. [7]

Finalmente, de este estudio podemos concluir, que hay una gran oferta de este tipo de aplicaciones, pero que algunas de ellas todavía no tienen su versión para smartwatch. También podemos comprobar que la mayoría de las aplicaciones monitorizan y registran la actividad del usuario, para poder llevar un control de la actividad física que se realiza. Incluso algunas han creado una comunidad en la que los usuarios comparten entrenamientos y logros. En nuestro caso nos vamos a centrar más en el entrenamiento en sí, en que el usuario pueda configurarlo a su gusto y principalmente en que pueda realizar entrenamientos tanto por intervalos como normales.

Hemos realizado la siguiente tabla donde vemos que funcionalidades, las que más importancia hemos dado en nuestro proyecto, tienen las aplicaciones para smartwatch que hemos visto y los relojes Polar y Garmin:

Aplicación/ reloj deportivo	Entrenamiento por intervalos (Tiempo y distancia)	Configurar alertas	Configurar pantalla entrenamiento	Muestra gráfica frecuencia cardiaca	Muestra ruta de entrenamiento
Strava	NO	NO	NO	NO	NO
Runstatic	NO (aplicación móvil sí)	NO	SI	NO	NO
Garmin	SI	SI	SI	SI	NO
Polar	SI (configuración desde el móvil)	SI	SI	SI	NO
RunKeeper	NO	NO	NO	NO	NO
Google fit	NO	SI	SI	NO	NO

Tabla 1. Aplicaciones deportivas y funcionalidades.

3 Diseño

3.1 Cuestionario

Antes de empezar con los requisitos, se ha realizado un formulario mediante *Google Forms*, el cual se puede ver en el anexo B, con el que hemos querido recopilar información y opiniones de usuarios. El formulario está dividido en varias secciones:

- Introducción: se le expone al usuario de que se trata el proyecto que se va a realizar.
- Datos personales: Registramos algunos datos personales del usuario, incluyendo el correo electrónico para futuras cuestiones.
- Hábitos de entrenamiento: Se quiere comprobar que el usuario entrena con frecuencia.
- Equipamiento: Se quiere comprobar que los usuarios hacen uso de smartwatchs o de otro aparato para monitorizar su actividad física.
- Sistema de monitorización: Se realizan una serie de preguntas sobre las funcionalidades más interesantes que creemos que puede tener nuestra aplicación, para saber la opinión del usuario sobre ellas.
- La última sección es un campo de texto por si algún usuario ha echado algo importante en falta o quiere recalcar algo.

Lo primero de todo ha sido realizar un plan de difusión, en el que hemos querido que los usuarios que respondiesen el cuestionario practicasen deporte con frecuencia y a ser posible que hicieran uso de algún sistema para monitorizar su actividad, principalmente smartwatchs. Teniendo claro esto, mi tutor, Javier Gómez Escribano, contacto con un compañero doctor de INEF, mediante el cual conseguimos unas cuantas respuestas. Después yo a través de mi cuñado, que entrena desde hace años con un grupo de personas con las que se prepara para realizar pruebas como maratones, triatlones, …. También conseguimos recibir algunas respuestas. Lo máximo que hemos podido obtener han sido 38 respuestas [8] y [9], pero podemos decir que son de buena calidad. Se intentó contactar con clubs como, Agua y verde, que es un club de triatlón, pero sin éxito.

Para comprobar que las respuestas que hemos recibido tienen validez para nuestro proyecto, realizamos la media de sesiones de entrenamiento que realizan los usuarios a la semana, la media de la duración de estas sesiones y contamos cuantos usuarios utilizan smartwatchs para entrenar y cuantos no:

- Media sesiones a la semana: 4.27
- Duración media de las sesiones: 57.7 minutos
- En la Ilustración 0-9 del anexo B se puede ver como casi el 70% de los usuarios sí que utilizan smartwatch.

Posteriormente se realiza un análisis de las respuestas de los usuarios sobre las preguntas relacionadas con las funcionalidades que proponemos. A continuación, se enumera las funcionalidades junto a las ilustraciones del anexo B en las que se puede apreciar que por lo general el usuario da el visto bueno a cada una de ellas:

- Entrenamiento por intervalos de tiempo o distancia. Ilustración 0-10.
- Entrenamiento por intervalos a una frecuencia cardiaca. Ilustración 0-11.
- Imponer ritmo al entrenamiento. Ilustración 0-12.

- Mostrar una gráfica de la frecuencia cardiaca del usuario durante el entrenamiento al finalizar este. Ilustración 0-13.
- Configurar alertas durante el entrenamiento. Ilustración 0-14.
- Feedback del entrenamiento. Ilustración 0-15.
- Historial y resumen de los entrenamientos realizados. Ilustración 0-16.
- Elegir la información que se quiere mostrar en la pantalla durante el entrenamiento. Ilustración 0-17.
- Guardar ruta de entrenamiento. Ilustración 0-18.

3.2 Análisis Requisitos

Después de haberse realizado el cuestionario y el análisis de sus respuestas, el análisis de algunas de las aplicaciones del mercado y haber fijado unos objetivos, se van a detallar todos los requisitos que necesita la aplicación, tanto funcionales como no funcionales, que engloben todas las funcionalidades de la aplicación.

3.2.1 Requisitos funcionales

- RF_1 : La aplicación debe permitir al usuario indicar su edad, altura y peso.
- RF_2: La aplicación debe calcular las zonas cardiacas del usuario
- RF_3: La aplicación debe medir la frecuencia en reposo del usuario.
- RF_4: La aplicación debe contar con un menú para facilitar la navegación.
- RF_5: La aplicación debe permitir al usuario consultar el historial de los entrenamientos realizados.
- RF_6: La aplicación debe permitir ver las estadísticas y los datos de los entrenamientos que el usuario tiene en su historial.
- RF_7: La aplicación debe permitir escoger al usuario el entrenamiento que quiere realizar entre estos deportes: bicicleta, running y natación.
- RF_8: La aplicación debe permitir al usuario realizar tanto entrenamientos libres como por intervalos.
- RF_9: La aplicación debe permitir al usuario decidir el número de intervalos que quiere que tenga el entrenamiento y si son iguales o diferentes.
- RF_10: La aplicación debe permitir al usuario introducir un calentamiento previo o un enfriamiento posterior al entrenamiento por intervalos.
- RF_11: La aplicación debe permitir al usuario indicar el tipo y la duración de cada intervalo.
- RF_12: La aplicación debe permitir al usuario indicar el tipo de descanso entre intervalos y su duración.
- RF_13: La aplicación debe permitir imponer una intensidad a cada intervalo.
- RF_14: La aplicación debe permitir añadir alarmas al entrenamiento.
- RF_15: La aplicación debe permitir indicar el tipo y la duración de las alarmas.
- RF_16: La aplicación debe permitir eliminar alarmas.
- RF_17: La aplicación debe permitir al usuario escoger los campos que quiere que se muestren durante el entrenamiento.
- RF_18: La aplicación debe permitir empezar el entrenamiento al usuario.
- RF_19: La aplicación debe permitir al usuario parar y reanudar el entrenamiento.
- RF_20: La aplicación debe permitir al usuario cancelar el intervalo activo durante el entrenamiento.

- RF_21: La aplicación debe permitir cancelar o reanudar las alarmas durante el entrenamiento.
- RF_22: La aplicación debe permitir al usuario finalizar el entrenamiento.
- RF_23: La aplicación debe notificar las alarmas que el usuario haya configurado.
- RF_24: La aplicación debe notificar el fin y el comienzo de un intervalo.
- RF_25: La aplicación al finalizar el entrenamiento debe mostrar las estadísticas al usuario.
- RF_26: La aplicación debe permitir al usuario iniciar se sesión en *Google* y aceptar que la aplicación almacene y lea datos en *Google Fit*.
- RF_27: La aplicación debe permitir al usuario activar los permisos de ubicación y sensores.
- RF_28: La aplicación debe interactuar con la API sensores de *Google Fit* para mostrar los datos necesarios al usuario.
- RF_29: La aplicación debe permitir al usuario ver sus logros en cada uno de los deportes. (Mayor distancia recorrida, Mejor ritmo, Mayor número de calorías consumidas)
- RF_30: La aplicación debe saber la localización del usuario durante el entrenamiento.
- RF_31: La aplicación debe poder tomar medidas de las constantes vitales del usuario (Frecuencia cardíaca).

3.2.2 Requisitos no funcionales

- RFN_1: Sistema operativo Android Wear 2.0 o posterior.
- RFN_2: La aplicación está pensada para smartwatchs que cumplan el requisito anterior en cuanto a su sistema operativo.
- RFN_3: El usuario interactuará con la aplicación mediante la pantalla táctil del smartwatch.
- RFN_4: Las alertas que tenga que mostrar la aplicación al usuario se mostrarán a partir de las notificaciones de Android.
- RFN_5: La aplicación no necesitará conexión a Internet para su uso.
- RFN_6: Esta aplicación será únicamente para smartwatch, y no necesitará de un Smartphone para su funcionamiento.
- RFN_7: La aplicación está pensada para smartwatchs que cuenten con GPS y pulsómetro.

3.3 Casos de uso

En esta sección se va a describir los pasos o las actividades que vamos a tener que realizar para llevar a cabo las diferentes funcionalidades de las que consta la aplicación. Para ello en primer lugar se identifican y definen los actores que participan en las funcionalidades. Posteriormente se crea un diagrama en el que se especificarán la relación de los actores con los casos de uso. Finalmente se enumerarán y describirán cada uno de los casos de uso.

3.3.1 Definición de actores del sistema

Se llama actor a una entidad externa que interactúa con el sistema, puede ser una persona física o un sistema externo que participe en algún caso de uso. En la aplicación identificamos los siguientes actores:

- **Usuario**: Cualquier persona que hace uso de la aplicación.
- **Google Fit**: Una herramienta de almacenamiento y lectura de datos fitness externa a la aplicación.

3.3.2 Diagrama de casos de uso

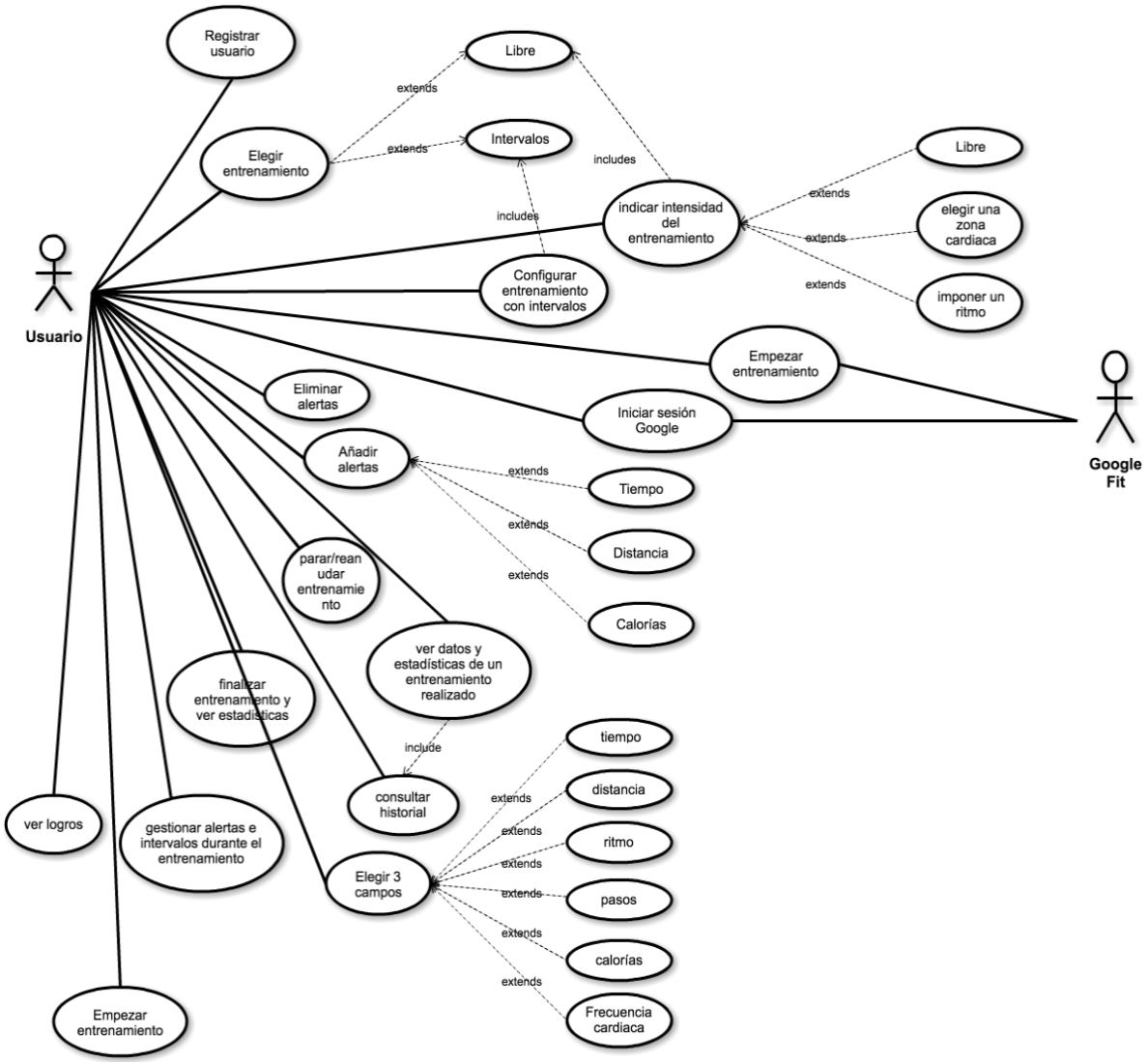

Ilustración 3-1. Diagrama de casos de uso de la aplicación

3.3.3 Descripción de casos de uso

En las siguientes tablas se describen detalladamente algunos de los casos de uso del diagrama anterior. Para ello he utilizado la guía de redacción de casos de uso de MADEJA (Marco de Desarrollo De la Junta de Andalucía) [10].

CU-01	Iniciar sesión en Google	
Versión	20-2-2018	
Precondición	El usuario accede a la aplicación por primera vez.	
Descripción	El sistema deberá pedir al usuario que inicie sesión en google.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accede a la aplicación.
	2	Al usuario se le pide que inicie sesión con su cuenta de google.
	3	El usuario indica su cuenta de google e inicia sesión.
	4	Al usuario se le pide que permita a la aplicación leer y almacenar datos en Google Fit.
	5	El usuario da permiso a la aplicación.
Post-condición	El usuario ya está registrado en la aplicación y se calcula su frecuencia cardiaca máxima y sus zonas cardíacas.	
Excepciones	3	El inicio de sesión en Google falla. Se le comunica al usuario y se retorna al paso 2.
Comentarios		

Tabla 2. Caso de uso CU-01, Iniciar sesión con Google

CU-02	Registrar usuario	
Versión	20-2-2018	
Dependencias		
Precondición	El usuario ha accedido por primera vez a la aplicación y ya ha iniciado sesión en google.	
Descripción	El sistema deberá realizar los pasos que se explican a continuación cuando un usuario accede a la aplicación por primera vez.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema solicita al usuario que indique su edad, peso y altura.
	2	El usuario introduce los datos solicitados
	3	El sistema solicita al usuario permiso para acceder a los datos de los sensores de sus constantes vitales
	4	El usuario da permiso a la aplicación.
	5	El sistema comunica al usuario que se va a proceder a medir su frecuencia cardiaca en reposo.
	6	El usuario acepta y se procede a la medición
	7	El sistema finaliza la medición y registra al usuario.
Post-condición	El usuario ya esta registrado en la aplicación y se calcula su frecuencia cardiaca máxima y sus zonas cardíacas.	
Excepciones		
Comentarios		

Tabla 3. Caso de uso CU-02, Registrar usuario

CU-03	Elegir entrenamiento	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha accedido a la aplicación	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	El sistema le muestra al usuario el menú principal de la aplicación
	2	El usuario elige la opción *TRAINING*
	3	El sistema le muestra los deportes que puede entrenar.
	4	El usuario elige uno de los deportes.
	5	El sistema le da la opción al usuario de elegir un entrenamiento libre o por intervalos.
	6	El usuario elige uno de los tipos de entrenamiento.
Post-condición		
Excepciones		
Comentarios		

Tabla 4. Caso de uso CU-03, Elegir entrenamiento

CU-04	Indicar intensidad del entrenamiento	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha elegido un entrenamiento libre.	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	El sistema le da a elegir al usuario entre una intensidad libre, indicar una zona cardiaca o imponer un ritmo.
	2	El usuario elige una de las opciones que le da la aplicación.
Post-condición		
Excepciones		
Comentarios		

Tabla 5. Caso de uso CU-04, Indicar intensidad del entrenamiento

CU-05	Configurar entrenamiento con intervalos	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha elegido practicar un entrenamiento por intervalos.	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	Al usuario se le muestra las diferentes secciones que puede configurar.
	2	El usuario elige indicar el número de repeticiones y si estas van a ser iguales o diferentes.
	3	El sistema le muestra una pantalla para que el usuario configure las repeticiones

	4	El usuario elige indicar el tipo y la duración de los intervalos.
	5	El sistema muestra una pantalla para que el usuario configure el intervalo.
	6	El usuario añade un calentamiento al entrenamiento.
	7	El usuario añade un enfriamiento al entrenamiento.
	8	El usuario elige indicar la intensidad de los intervalos.
	9	El sistema muestra una pantalla para la configuración de la intensidad.
Post-condición		
Excepciones		
Comentarios		

Tabla 6. Caso de uso CU-05, Configurar entrenamiento con intervalos

CU-06	**Añadir avisos**	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha elegido un entrenamiento.	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	El sistema le muestra al usuario un menú con la opción de configurar el entrenamiento o de empezar el entrenamiento.
	2	El usuario selecciona la opción congifurar.
	3	El sistema le muestra al usuario la opción de elegir añadir una alerta o seleccionar los campos.
	4	El usuario selecciona la opción de añadir una alerta.
	5	El sistema le muestra una pantalla para indicar el tipo y la duración de la alerta.
	6	El sistema añade la alerta y le muestra las alertas que ha añadido.
Post-condición		
Excepciones		
Comentarios		

Tabla 7. Caso de uso CU-06, Añadir avisos

CU-07	**Eliminar avisos**	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha añadido al menos una alerta.	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	Selecciona la opción alertas en el menú de configuración del entrenamiento.
	2	El sistema muestra las alertas que el usuario ha añadido al entrenamiento.

	3	El usuario presiona la alerta que quiere eliminar.
	4	El sistema le muestra si quiere eliminar la alerta.
	5	El usuario selecciona que sí.
Post-condición		
Excepciones	6	El usuario decide finalmente no cancelar la alerta.
Comentarios		

Tabla 8. Caso de uso CU-07, Eliminar avisos

CU-08	**Elegir 3 campos**	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha elegido un entrenamiento.	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	El sistema le muestra al usuario un menú con la opción de configurar el entrenamiento o de empezar el entrenamiento.
	2	El usuario selecciona la opción de configurar.
	3	El sistema le muestra al usuario la opción de añadir una alerta o seleccionar los campos.
	4	El usuario selecciona la opción de seleccionar campos.
	5	El sistema le muestra al usuario los 3 campos que se van a mostrar por pantalla.
	6	El usuario selecciona uno de los campos para cambiarlo.
	7	El sistema le muestra al usuario todos los tipos de campos que puede mostrar.
	8	El usuario elige uno de los campos.
Post-condición		
Excepciones		
Comentarios		

Tabla 9. Caso de uso CU-08, Elegir 3 campos

CU-09	**Empezar entrenamiento**	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha elegido un entrenamiento libre y ha indicado su identidad o el usuario ha elegido un entrenamiento por intervalos y lo ha configurado.	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	El sistema muestra al usuario la opción de empezar el entrenamiento o configurarlo.
	2	El usuario elige la opción de empezar el entrenamiento.
	3	El sistema comienza a cronometrar el tiempo y a interactuar con Google Fit para monitorizar la actividad.
Post-condición		

Excepciones	
Comentarios	

Tabla 10. Caso de uso CU-09, Empezar entrenamiento

CU-10	**Parar/reanudar entrenamiento**	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha elegido empezar el entrenamiento.	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	El sistema le muestra la opción de parar/reanudar el entrenamiento.
	2	El usuario elige la opción parar/reanudar el entrenamiento
	3	El sistema para el cronometro y la lectura de datos en Google Fit o reanuda el cronometro y la lectura de datos en Google Fit.
Post-condición		
Excepciones		
Comentarios		

Tabla 11. Caso de uso CU-10, Parar/reanudar entrenamiento

CU-11	**Gestionar alertas e intervalos durante el entrenamiento**	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha elegido empezar el entrenamiento	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	El sistema le muestra los intervalos del entrenamiento y las alertas
	2	El usuario mantiene pulsado la alerta o el intervalo que quiere cancelar.
	3	El sistema cancela el intervalo o deja de notificar la alerta.
Post-condición		
Excepciones		
Comentarios		

Tabla 12. Caso de uso CU-11, Gestionar alertas e intervalos durante el entrenamiento

CU-12	**Finalizar entrenamiento y ver estadísticas**	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha elegido empezar el entrenamiento	
Descripción		
Secuencia normal	**Paso**	**Acción**

	1	El sistema le muestra al usuario la opción de finalizar el entrenamiento.
	2	El usuario pincha en la opción finalizar el entrenamiento.
	3	El sistema finaliza el entrenamiento y muestra sus estadísticas al usuario.
Post-condición		
Excepciones		
Comentarios		

Tabla 13. Caso de uso CU-12, Finalizar entrenamiento y ver estadísticas

CU-13	**Consultar historial**	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha accedido a la aplicación	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	El sistema le muestra al usuario la opción de ver el historial.
	2	El usuario pincha en la opción de ver el historial
	3	El sistema le muestra al usuario los entrenamientos que haya realizado.
Post-condición		
Excepciones		
Comentarios		

Tabla 14. Caso de uso CU-13, Consultar historial

CU-14	Ver datos y estadísticas de un entrenamiento realizado	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha accedido a su historial.	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	El sistema le muestra al usuario los entrenamientos que ha realizado.
	2	El usuario pincha en el entrenamiento del cual quiere ver sus datos y sus estadísticas.
	3	El sistema le muestra al usuario los datos y las estadísticas del entrenamiento que ha elegido.
Post-condición		
Excepciones		
Comentarios	En caso de que el usuario no haya realizado ningún entrenamiento se le mostrará una lista vacía.	

Tabla 15. Caso de uso CU-14, ver datos y estadísticas de un entrenamiento realizado

CU-15	Ver logros	
Versión	4-2-2018	
Dependencias		
Precondición	El usuario ha accedido a la aplicación	
Descripción		
Secuencia normal	**Paso**	**Acción**
	1	El sistema le muestra al usuario la opción de ver los logros
	2	El usuario pincha en la opción de ver logros.
	3	El sistema le muestra al usuario sus logros.
Post-condición		
Excepciones		
Comentarios	En caso de que el usuario no haya realizado ningún entrenamiento no tendrá ningún logro.	

Tabla 16. **Caso de uso CU-15, Ver logros**

3.3.4 Matriz de trazabilidad de casos de uso

En la siguiente tabla se muestra los casos de uso que cubren cada requisito funcional, donde se puede comprobar que todos los requisitos están cubiertos mediante el diseño realizado.

RF/CU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RF_1		x													
RF_2		x													
RF_3		x													
RF_4			x										x		x
RF_5													x		
RF_6														x	
RF_7			x												
RF_8			x												
RF_9					x										
RF_10					x										
RF_11					x										
RF_12					x										
RF_13					x										
RF_14						x									
RF_15						x									
RF_16							x								
RF_17								x							
RF_18									x						
RF_19										x					
RF_20											x				
RF_21											x				
RF_22												x			
RF_23											x				
RF_24											x				
RF_25												x			
RF_26	x														
RF_27		x													
RF_28									x						
RF_29															x
RF_30		x													
RF_31		x													

Tabla 17. Matriz de trazabilidad

3.4 Interfaz gráfica

Se diseñan una serie de plantillas que muestran como sería el aspecto de la aplicación. Este diseño se ha realizado mediante el programa *InDesign* de Adobe, al no encontrar ningún programa realmente válido para crear plantillas para smartwatch. A continuación, se muestran las plantillas junto a una pequeña descripción de cada una de ellas

3.4.1 Maquetas

Esta sería la primera pantalla, ilustración 3-2, que veríamos en la aplicación. Un *splashscreen* con el logo de la aplicación. A continuación, en caso de que el usuario iniciase la aplicación por primera vez, tendría que indicar en la ilustración 3-3, su peso, su edad y su altura, para registrar estos datos en la aplicación.

Ilustración 3-2. Maqueta inicio.

Ilustración 3-3. Maqueta datos usuario.

Al igual que la anterior pantalla, las siguientes pantallas sólo aparecerán en caso de que el usuario todavía no haya sido registrado en la aplicación. La pantalla correspondiente a la ilustración 3-4, indica al usuario que se va a proceder a medir su frecuencia cardiaca en reposo y que debe estar relajado. Posteriormente cuando el usuario pinche en el botón la aplicación comenzará a medir su frecuencia en reposo en la ilustración 3-5, en la que el texto mostrará su frecuencia cardiaca en cada momento y el circulo amarillo es un *Progressbar* que indicará el tiempo que queda para que finalice el cálculo. Finalmente, una vez realizado el cálculo, en la ilustración 3-6 se mostrará al usuario el resultado.

Ilustración 3-5. Maqueta mensaje frecuencia en reposo

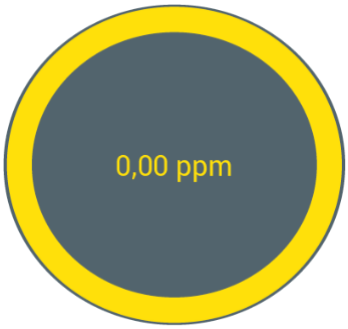

Ilustración 3-4. Maqueta cálculo frecuencia en reposo

Ilustración 3-6. Maqueta mostrar frecuencia en reposo

La siguiente pantalla, ilustración 3-7, sería la siguiente en caso de que el usuario todavía no estuviese registrado en la aplicación, en caso de ya estar registrado sería la primera pantalla después de la ilustración 3-2. Esta pantalla refleja el menú principal de la aplicación en la que el usuario puede elegir realizar un entrenamiento nuevo, ver las estadísticas de los entrenamientos realizados y ver sus mejores marcas, sus logros. Si el usuario elige la opción *TRAINING* pasará al menú de la ilustración 3-8 donde eligirá el deporte y finalmente en la ilustración 3-9, indicará el tipo de entrenamiento que quiere realizar.

Ilustración 3-7. Maqueta menú principal

Ilustración 3-9. Maqueta menú deportes

Ilustración 3-8. Maqueta menú tipo entrenamiento

Si el usuario elige realizar un entrenamiento *LIBRE*, a continuación indicará en la ilustración 3-10, a que intensidad quiere entrenar. En caso de que elija *Z CARDIACAS*, la siguiente ilustración sería la 3-11, para que el usuario escogiese una de las zonas cardiacas. Estas ilustraciones se repetirían a la hora de elegir una internsidad para un intervalo.

Ilustración 3-10. Maqueta menú tipo intensidad.

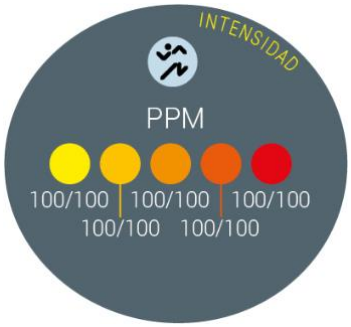

Ilustración 3-11. Maqueta zonas cardiacas.

Por otro lado si el usuario eligiese realizar un entrenamiento por intervalos, accedería ha esta pantalla, donde iría pinchando en las diferentes secciones para configurar su entrenamiento por intervalos.

Ilustración 3-12. Maqueta configuración intervalos

Si el usuario pincha en la sección de *REPETICIONES*, accedería a la siguiente pantalla, ilustración 3-13, en la que decidiría si quiere que las repeticiones sean iguales o diferentes, y el número de repeticiones del entrenamiento. Por otro lado, si el usuario pincha en la sección *INTERVALO*, y ha elegido que las repeticiones sean iguales, accedería a la ilustración 3-14, donde decidirá de que tipo será el intervalo y su duración (Esto ocurrirá igual en la sección *DESCANSO*). Si el usuario ha elegido repeticiones diferentes, el usuario accedería a la ilustración 3-15, e iría escogiendo para cada intervalo su tipo y su duración.

Ilustración 3-14. Maqueta número y tipo de repeticiones

Ilustración 3-13. Maqueta luración y tipo de intervalos

Ilustración 3-15. Maqueta duración y tipo de intervalo

Una vez configurado el entrenamiento por intervalos o una vez escojida la intensidad de un entrenamiento libre, se accedería a la siguiente pantalla, ilustración 3-16, en la que puedes configurar el entrenamiento o darle a start, para que el entrenamiento comience. En caso de que elija la opción configurar se le mostrará la ilustración 3-17, donde puede elegir entre añadir una alerta o configurar los campos que quiere mostrar durante el entrenamiento.

Ilustración 3-16. Maqueta menú entrenamiento

Ilustración 3-17. Maqueta menú configuración

En caso de haber escogido añadir una alerta, si no se ha añadido ninguna alerta previamente, deberá indicar en la siguiente pantalla, ilustración 3-18, el tipo y duración de esa alerta. Pero en caso de que ya haya alertas añadidas se accederá directamente a la ilustración 3-18. Una vez el usuario ha añadido la alerta, se mostrarán las alertas que el usuario ha añadido, ilustración 3-19, en caso de querer añadir otra pinchará el boton + y en caso de no querer añadir más pinchará el ok. Si quiere eliminar alguna, presionará en ella y se le mostrará la ilustración 3-20.

Ilustración 3-20. Maqueta menú tipo y duración

Ilustración 3-19. Maqueta alertas añadidas

Ilustración 3-18. Maqueta eliminar alertas

Si el usuario quiere indicar que campos se van a mostrar durante el entrenamiento, se mostrará la ilustración 3-21. Y simplemente pinchando en alguno de los campos accederá al menú de la ilustración 3-22 para escoger el dato que quiere mostrar en ese campo.

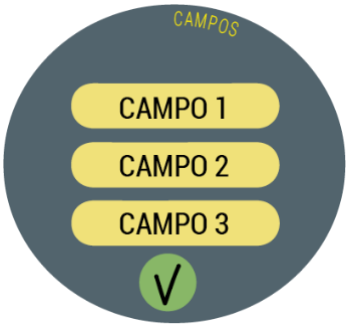

Ilustración 3-22. Maqueta configuración campos

Ilustración 3-21. Maqueta menú campos.

Si el usuario en la ilustración 3-16 pincha en *start*, se mostrarían ilustración 3-23 para un entrenamiento libre y la ilustración 3-24 para un entrenamiento por intervalos. Y la ilustración 3-25 nos permite parar y reanudar el entrenamiento, y cancelar las alertas y los intervalos en caso de que el entrenamiento los tenga.

Ilustración 3-25. Maqueta de entrenamiento libre

Ilustración 3-24. Maqueta de entrenamiento por

Ilustración 3-23. Maqueta de entrenamiento por

Cuando el usuario finaliza el entrenamiento se le mostrarán las estadísticas del entrenamiento. La ilustración 3-26 representa las estadisticas de un entrenamiento libre y la ilustración 3-27 representa las estadisticas de un entrenamiento con intervalos, que el usuario irá deslizando hacia abajo para ver las estadisticas de todos los intervalos:

Ilustración 3-26. Maqueta estadísticas entrenamiento

Ilustración 3-27. Maqueta estadísticas entrenamiento

Si el usuario pincha en *ESTADISTICAS* en el menu principal, ilustración 3-7, se le mostrará la siguiente pantalla, ilustración 3-28, con el historial de los entrenamientos que ha realizado. Cuando el usuario pinche en alguno de los entrenamientos de la pantalla anterior se le mostrará los datos del entrenamiento y si desliza horizontalmente el dedo podrá ver las estadísticas, ilustraciones 3-29 y 3-30.

Ilustración 3-29. Maqueta historial entrenamientos

Ilustración 3-30. Maqueta datos del entrenamiento

Ilustración 3-28. Maqueta estadísticas entrenamiento

Si el usuario pincha en *LOGROS* en el menu principal, ilustración 3-7, se le mostrará la siguiente pantalla con los logros del usuario. Deslizando el dedo horizontalmente veremos los logros de cada deoporte:

Ilustración 3-31. Maqueta logros

3.5 Modelado de datos

En esta sección se va a mostrar uno de los pasos más importantes del diseño, realizar un diseño del modelo de información minuciosamente. Teniendo en cuenta que tiene que cumplir los requisitos enunciados anteriormente. Este modelo tendrá que tener una buena estructura para intentar garantizar un diseño definitivo, pero que en caso de que se produzca un cambio, este no provoque una alteración importante en él, ya que al realizar un desarrollo iterativo e incremental podría suceder.
El modelo debe tener en cuenta los siguientes requisitos obtenidos anteriormente:

- Los datos de localización y de constantes vitales, serán obtenidos a través de un servidor externo de Google (Google Fit), que serán registrados en la aplicación para permitir al usuario su visualización.
- Del usuario se registrará su información básica, y algunos datos que se calcularán en la aplicación necesarios para realizar algunos entrenamientos personalizados (frecuencia cardiaca en reposo y zonas de frecuencia cardiaca, explicado en el anexo D)

- Los entrenamientos deben permitir registrar una gran cantidad de datos, ya que pueden llegar a ser muy complejos (entrenamiento por intervalos).

A continuación, se muestra el modelo de datos en la siguiente ilustración:

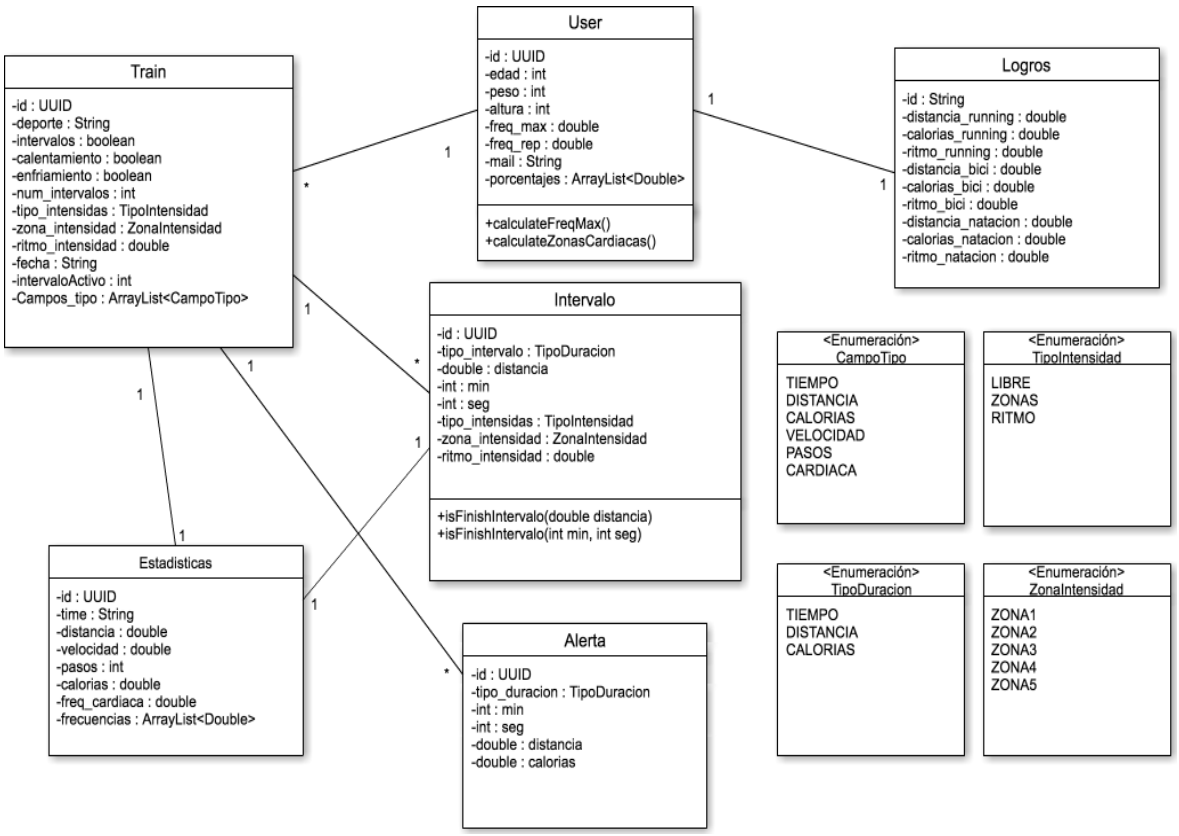

Ilustración 3-32. Diagrama de clases UML de la aplicación

A continuación, se describirán cada una de las clases que se representan en el diagrama, sus atributos y sus relaciones:

3.5.1 Enumeración *CampoTipo*

La enumeración *CampoTipo* representa los diferentes tipos de campos que el usuario puede escoger para mostrar en la pantalla del smartwatch durante el entrenamiento:

- **TIEMPO**: indica el tiempo que transcurre.
- **DISTANCIA**: indica la distancia que recorre el usuario.
- **CALORIAS**: indica las calorías que consume el usuario.
- **VELOCIDAD**: indica a la velocidad que va el usuario.
- **PASOS**: indica los pasos que da el usuario.
- **CARDIACA**: indica la frecuencia cardiaca del usuario.

3.5.2 Enumeración *TipoIntensidad*

La enumeración *TipoIntensidad* representa los diferentes tipos de intensidad a los que el usuario puede entrenar:

- **LIBRE**: indica una intensidad a gusto del usuario.
- **ZONAS**: indica que el usuario quiere entrenar entre las frecuencias cardiacas de una de las zonas cardiacas del usuario.
- **RITMO**: indica que el usuario quiere imponer un ritmo al entrenamiento.

3.5.3 Enumeración *TipoDuracion*

La enumeración *TipoDuracion* representa los diferentes tipos de duración que pueden tener los intervalos o cada cuanto se notificará una alerta:

- **TIEMPO**: indica que la duración de un intervalo es de tiempo o que una alerta se notificará después de un tiempo.
- **DISTANCIA**: indica que la duración de un intervalo es de distancia o que una alerta que se notificará después de una distancia recorrida por el usuario.
- **CALORIAS**: indica una alerta que se notificará después de unas calorías consumidas por el usuario.

3.5.4 Enumeración *ZonaIntensidad*

La enumeración *ZonaIntensidad* representa las diferentes zonas cardiacas a las que puede entrenar el usuario:

- **ZONA1**: representa la zona cardiaca entre el 50% y el 60% de la frecuencia cardiaca máxima del usuario.
- **ZONA2**: representa la zona cardiaca entre el 60% y el 70% de la frecuencia cardiaca máxima del usuario.
- **ZONA3**: representa la zona cardiaca entre el 70% y el 80% de la frecuencia cardiaca máxima del usuario.
- **ZONA4**: representa la zona cardiaca entre el 80% y el 90% de la frecuencia cardiaca máxima del usuario.
- **ZONA5**: representa la zona cardiaca entre el 90% y el 100% de la frecuencia cardiaca máxima del usuario.

3.5.5 Clase *User*

La clase *User*, representa al usuario que utiliza la aplicación. Tiene los siguientes atributos:

- ***Id***: Es un tipo UUID (Universally Unique Identifier) del paquete java.util.UUID, que se generará mediante el método randomUUID, que genera identificadores únicos de manera aleatoria.
- ***Edad, peso y altura***; Son tres int que representan la edad, el peso y la altura, que serán necesarios para poder obtener la frecuencia cardiaca máxima del usuario y para obtener las calorías que consume el usuario durante el entrenamiento.
- ***Freq_max***: Es un tipo double que representa la frecuencia cardiaca máxima del usuario.
- ***Freq_rep***: Es un tipo double que representa la frecuencia en reposo del usuario.

- ***Mail***: Es un tipo String que representa la cuenta de google con la que inicia sesión el usuario.
- ***Porcentajes***: Es un ArrayList de tipo Double que representa los porcentajes entre los que se encuentras las zonas cardiacas del usuario.

La clase *User* dispone de las siguientes relaciones:

- **Relación con la clase *Train***: almacena los entrenamientos que ha realizado el usuario.
- **Relación con la clase *Logros***: almacena los logros del usuario.

3.5.6 Clase *Train*

La clase *Train*, representa los entrenamientos que el usuario realiza en la aplicación. Tiene los siguientes atributos:

- ***Id***: igual que el id de User.
- ***Deporte***: Es un tipo String que representa el deporte que el usuario va a entrenar.
- ***Intervalos***: Es un tipo boolean que representa si el entrenamiento tiene o no tiene intervalos.
- ***Calentamiento y enfriamiento***: Son dos tipos boolean que representan si un entrenamiento por intervalos va a tener un calentamiento previo y un enfriamiento posterior.
- ***Num_intervalos***: Es un int que representa el número de intervalos que tiene un entrenamiento.
- ***Tipo_intensidad***: Es un TipoIntensidad que representa el tipo de intensidad del entrenamiento.
- ***Zona_intensidad***: Es un ZonaIntensidad que representa la zona cardiaca en la que el usuario quiere entrenar.
- ***Ritmo_intensidad***: Es un double que representa el ritmo impuesto por el usuario para el entrenamient**o.**
- ***Fecha***: Es un String que representa la fecha en la que se realizó el entrenamiento.
- ***Intervalo_activo***: Es un tipo int que representa el intervalo en curso del entrenamiento.
- ***Campos_tipo***: Es un ArrayList de CampoTipo que representa los campos que el usuario quiere que se muestren en la pantalla del smartwatch durante el entrenamiento.

La clase *Train* dispone de las siguientes relaciones:

- **Relación con la clase *Intervalo***: almacena los intervalos que tiene un entrenamiento en caso de tenerlos.
- **Relación con la clase *Estadísticas***: almacena las estadísticas del entrenamiento.
- **Relación con la clase *Alerta***: almacena las alertas que ha configurado el usuario para el entrenamiento.

3.5.7 Clase *Intervalo*

La clase *Intervalo*, representa los intervalos que tiene un entrenamiento. Tiene los siguientes atributos:

- ***Id***: igual que el id de User.
- ***Tipo_intervalo***: Es un TipoDuracion que representa el tipo de duración del intervalo.

- ***Distancia***: Es un double que en caso de que el tipo de duración del intervalo sea distancia, representa la distancia del intervalo.
- ***Min y seg***: Son dos int que en caso de que el tipo de duración del intervalo sea tiempo, representa el tiempo que dura el intervalo.
- ***Tipo_intensidad***: Es un TipoIntensidad que representa el tipo de intensidad del intervalo.
- ***Zona_intensidad***: Es un ZonaIntensidad que representa la zona cardiaca en la que el usuario quiere entrenar.
- ***Ritmo_intensidad***: Es un double que representa el ritmo impuesto por el usuario para el intervalo**.**

La clase *Intervalo* dispone de las siguientes relaciones:

- **Relación con la clase *Estadisticas***: almacena las estadísticas del intervalo.

3.5.8 Clase *Alerta*

La clase *Alerta*, representa las alertas que configura el usuario para que le avisen y notifiquen lo que quiera durante el entrenamiento. Tiene los siguientes atributos:

- ***Id***: igual que el id de User.
- ***Tipo_duracion***: Es un TipoDuracion que representa el tipo de duración de la alerta.
- ***Distancia***: Es un double que en caso de que el tipo de duración de la alerta sea distancia, representa cada cuanta distancia recorrida por el usuario se notificará al usuario la alerta.
- ***Min y seg***: Son dos int que en caso de que el tipo de duración de la alerta sea tiempo, representan cada cuanto tiempo transcurrido se notificará al usuario la alerta.
- ***Calorias***: Es un double que en caso de que el tipo de duración de la alerta sea calorías, representa cada cuantas calorías consumidas por el usuario se notificará la alerta.

3.5.9 Clase *Estadisticas*

La clase *Estadisticas*, representa las estadísticas finales de un entrenamiento o un intervalo al finalizar. Tiene los siguientes atributos:

- ***Id***: igual que el id de User.
- ***Time***: Es un String que representa el tiempo que ha durado el entrenamiento o el intervalo.
- ***Distancia***: Es un double que representa la distancia que el usuario ha recorrido durante un entrenamiento o un intervalo.
- ***Velocidad***: Es un double que representa la velocidad media del usuario durante un entrenamiento o un intervalo.
- ***Pasos***: Es un int que representa los pasos que el usuario ha dado durante un entrenamiento o un intervalo.
- ***Calorias***: Es un double que representa las calorías consumidas por el usuario durante un entrenamiento o un intervalo.
- ***Freq_cardiaca***: Es un double que representa la frecuencia cardiaca media del usuario durante un entrenamiento o un intervalo.
- ***Frecuencias***: Es un ArrayList de Double que representa las frecuencias que se han ido registrando durante el entrenamiento o un intervalo.

3.5.10 Clase Logros

La clase *Logros*, representa los logros de un usuario en cada uno de los deportes que se puede entrenar en la aplicación. Tiene los siguientes atributos:

- ***Id***: igual que el id de User.
- ***Distancia_running***: Es un double que representa la mayor distancia recorrida corriendo por el usuario.
- ***Calorias_running***: Es un double que representa el mayor número de calorías consumidas corriendo por el usuario.
- ***Ritmo_running***: Es un double que representa el mejor ritmo que un usuario ha tenido corriendo.
- ***Distancia_bici***: Es un double que representa la mayor distancia recorrida en bici por el usuario.
- ***Calorias_bici***: Es un double que representa el mayor número de calorías consumidas en bici por el usuario.
- ***Ritmo_bici***: Es un double que representa el mejor ritmo que un usuario ha tenido en bici.
- ***Distancia_natacion***: Es un double que representa la mayor distancia recorrida nadando por el usuario.
- ***Calorias_natacion***: Es un double que representa el mayor número de calorías consumidas nadando por el usuario.
- ***Ritmo_natacion***: Es un double que representa el mejor ritmo que un usuario ha tenido nadando.

4 Desarrollo

En esta sección se van a explicar algunas de las partes más importantes del desarrollo de nuestra aplicación. Antes de ello ha sido realizado el análisis y el diseño como se puede comprobar en los apartados anteriores.

Para la implementación de la aplicación se ha seguido el patrón de arquitectura de software MVC, que separa la lógica y las vistas de la aplicación. En este patrón se pueden diferenciar 3 componentes, separados entre sí, lo que nos garantiza que un cambio en una parte de nuestro código no supone un cambio en otra parte del mismo. A continuación, se define la responsabilidad de cada uno de los componentes:

- El modelo: Representa la información con la que la aplicación trabaja. Gestiona el acceso y la actualización de esa información.
- La vista: Representa la interfaz de usuario.
- El controlador: Es el componente de unión entre los otros dos, se encarga de recibir las órdenes del usuario, solicitar los datos al modelo y compartirlos a la vista.

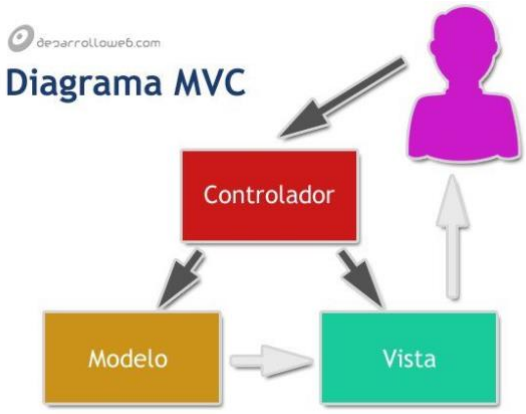

Ilustración 4-1

4.1 Base de datos

En primer lugar, se va a realizar un diseño de la base de datos que vamos a utilizar para almacenar toda la información necesaria para el correcto funcionamiento de la aplicación. En un principio no se iba a almacenar en la base de datos los entrenamientos y sus estadísticas, ya que como se explica en el anexo C, la API de Google Fit permite almacenar y leer datos de entrenamiento del usuario en la nube con facilidad, pero como en la aplicación se puede configurar un entrenamiento por intervalos tan al detalle, las sesiones de entrenamiento que un usuario puede crear en la aplicación no serían registradas correctamente en la API de Google Fit. En ella se puede registrar una sesión de entrenamiento incluso con repeticiones(intervalos), pero no se puede registrar que tipo de duración tiene el intervalo ni su duración en sí, este fue el motivo principal por lo que nos decantamos finalmente por almacenar toda la información en una base de datos.
Esta base de datos tiene el siguiente diseño:

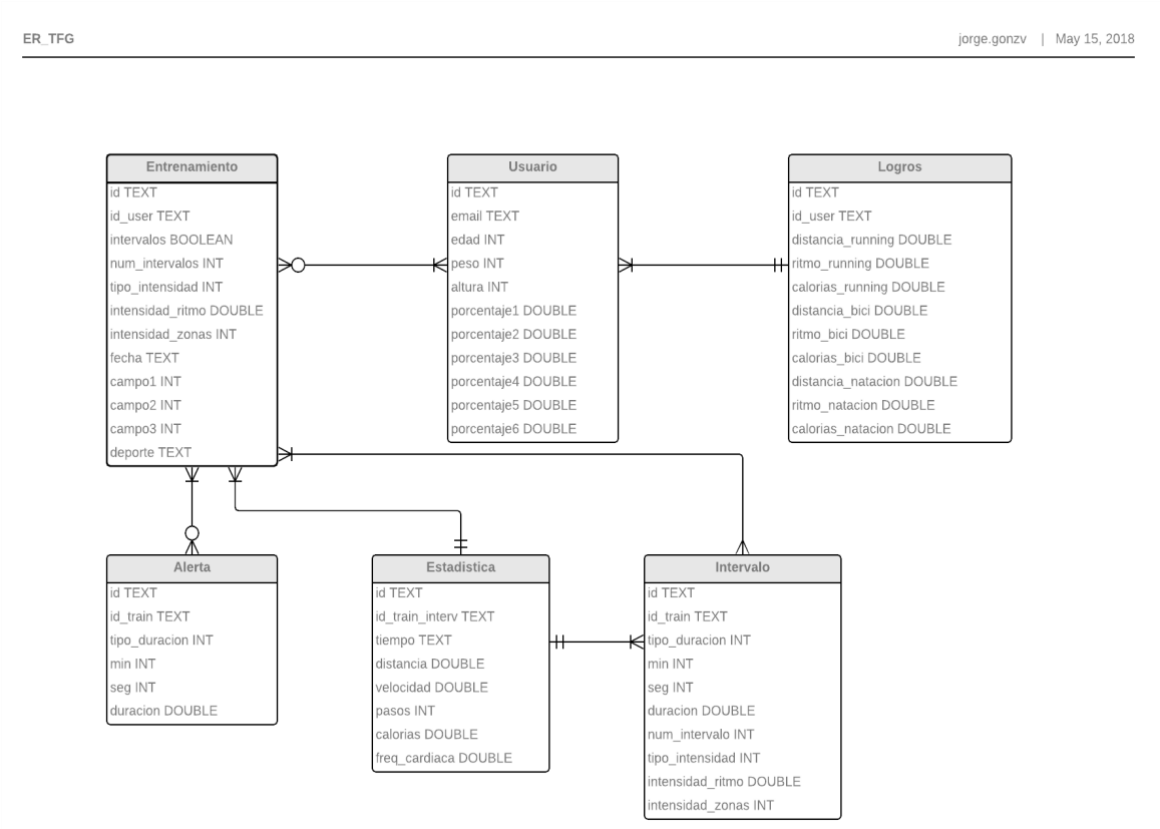

Ilustración 4-2. Diagrama Entidad Relación de la base de datos.

Como se puede ver en el diagrama anterior, podemos ver representadas todas las clases que se diseñaron en el diseño, y se puede observar que su estructura es muy parecida al diagrama de clases de la ilustración 3-32.

4.2 Aplicación principal

En esta parte del desarrollo, se implementa las funcionalidades, de las cuales vamos a mostrar las partes más importantes. Para realizar el desarrollo de una de las partes fundamentales de la aplicación se ha utilizado la API de *Google Fit*, de la cual se ha hecho una explicación en el anexo C.

4.2.1 Aplicación *Standalone*

Lo primero vamos a mostrar el código que hay que añadir en el *AndroidManifest.xml*, para indicar que es una aplicación independiente de smartwatch, es decir, no necesita un Smartphone para su funcionamiento. [11]

```
<meta-data
    android:name="com.google.android.wearable.standalone"
    android:value="true" />
```

Ilustración 4-3

4.2.2 Inicio de sesión de google

En la aplicación es necesario que el usuario inicie sesión en su cuenta de google para poder leer y escribir datos fitness en la API de Google Fit. Lo primero que haremos es configurar el inicio de sesión para solicitar al usuario aparte de la información básica, el id de usuario y su email, ver y almacenar la información de la actividad realizada, los datos de ubicación y la información del sensor corporal en Google Fit. Para ello se ha implementado este código [12]:

```
private void googleConection() {
    GoogleSignInOptions options =
            new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                    .requestEmail()
                    .requestScopes(Fitness.SCOPE_BODY_READ, Fitness.SCOPE_ACTIVITY_READ_WRITE,
                            Fitness.SCOPE_LOCATION_READ_WRITE)
                    .build();

    GoogleSignInClient mSignInClient = GoogleSignIn.getClient( activity: this, options);
    Intent signInIntent = mSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, RC_SIGN_IN);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == RC_SIGN_IN) {
        Task<GoogleSignInAccount> task =
                GoogleSignIn.getSignedInAccountFromIntent(data);
        if (task.isSuccessful()) {
            // Sign in succeeded, proceed with account
            System.out.println("login con exito");
        } else {
            // Sign in failed, handle failure and update UI
            System.out.println("fallo en login" );
        }
    }
}
```

Ilustración 4-4. Inicio sesión Google

Cuando se ejecuta este código aparece la siguiente pantalla en el reloj para que el usuario pueda permitir a la aplicación ver y almacenar los datos que antes hemos mencionado en *Google Fit*. Esta petición de esos permisos se ven reflejadas en el código en esta línea:

```
.requestScopes(Fitness.SCOPE_BODY_READ, Fitness.SCOPE_ACTIVITY_READ_WRITE,
        Fitness.SCOPE_LOCATION_READ_WRITE)
```

Ilustración 4-5. Permisos *Google Fit*

Ilustración 4-6. Captura de pantalla permisos *Google Fit*

4.2.3 Permisos en tiempo de ejecución

Para permitir que la aplicación pueda acceder a la ubicación de nuestro smartwatch y a los datos de sensores de las constantes vitales del usuario (Ritmo cardíaco), tenemos que añadir en el archivo AndroidManifest.xml de nuestra proyecto las siguientes líneas:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.BODY_SENSORS" />
```

Ilustración 4-7. Permisos localización y sensores corporales

Con esto antes podría ser suficiente para que se nos garantice el acceso a los datos solicitados, pero ahora a partir de Android 6.0 (nivel de API 23), los usuarios conceden los permisos a las aplicaciones mientras se ejecutan y no cuando se instalan, por lo que es necesario añadir el siguiente código para garantizarse poder leer los datos que requiere tu aplicación. La solicitud de estos permisos no puede realizarse en la misma actividad, ya que solo se puede requerir un permiso por actividad.

El método hasRuntimePermission() comprueba si el permiso que requerimos ha sido ya permitido por el usuario. En caso de que no sea así llamaríamos al método requestRuntimePermissions() que solicitará al usuario que permita acceder a los datos que indiquemos.

```
private boolean hasRuntimePermissions(String permission) {
    int permissionState =
            ActivityCompat.checkSelfPermission( context: this, permission);
    return permissionState == PackageManager.PERMISSION_GRANTED;
}

private void requestRuntimePermissions(String permission) {
    boolean shouldProvideRationale =
            ActivityCompat.shouldShowRequestPermissionRationale(
                    activity: this, permission);

    if (shouldProvideRationale) {
        System.out.println("Displaying permission rationale to provide additional context.");
    } else {
        System.out.println("Requesting permission");
        ActivityCompat.requestPermissions(
                activity: freqActivity.this,
                new String[] {permission},
                GOOGLE_FIT_PERMISSIONS_REQUEST_CODE);
    }
}
```

Ilustración 4-8. Código solicitud de permisos en ejecución

```
@Override
public void onRequestPermissionsResult(
        int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    System.out.println("onRequestPermissionResult");
    if (requestCode == GOOGLE_FIT_PERMISSIONS_REQUEST_CODE) {
        if (grantResults.length <= 0) {
            // If user interaction was interrupted, the permission request is cancelled and you
            // receive empty arrays.
            System.out.println("User interaction was cancelled.");
        } else if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            // Permission was granted.
            System.out.println("PERMISO CONCEDIDO");
            heartSensor = new ControlerSensor( mActivity: freqActivity.this, DataType.TYPE_HEART_RATE_BPM, DataSource.TYPE_RAW,
            heartSensor.createFitnessFile();
            permission = true;
            return;
        } else {
            // Permission denied.
            System.out.println("permission denied");
            permission = false;
            return;
        }
    }
}
```

Ilustración 4-9. Código solicitud de permisos en ejecución 2

Con este método llamamos a los métodos anteriores para comprobar si el usuario ha permitido el acceso a los datos de los sensores de las constantes vitales, y en caso de no ser así se lo pediremos al usuario, lo cual hará que se muestre la siguiente pantalla en el reloj.

```
private void checkPermission() {
    if(hasRuntimePermissions(Manifest.permission.BODY_SENSORS)){
        System.out.println("El permiso ya esta concedido");
    }else{
        requestRuntimePermissions(Manifest.permission.BODY_SENSORS);
    }
}
```

Ilustración 4-10. Código comprobación permiso de sensores corporales

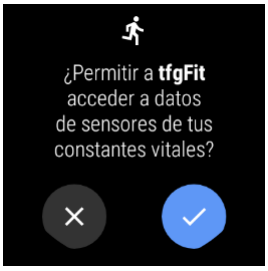

Ilustración 4-11. Captura solicitud de permiso de sensores corporales

Con este método llamamos a los métodos anteriores para comprobar si el usuario ha permitido el a la ubicación del dispositivo, y en caso de no ser así se lo pediremos al usuario, lo cual hará que se muestre la siguiente pantalla en el reloj.

```
private void permissionAcessFineLocation() {
    if (hasRuntimePermissions(Manifest.permission.ACCESS_FINE_LOCATION)) {
        return;
    } else {
        requestRuntimePermissions(Manifest.permission.ACCESS_FINE_LOCATION);
    }
}
```

Ilustración 4-12. Código comprobación permiso de localización

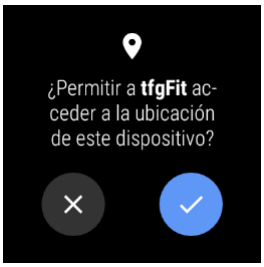

Ilustración 4-13. Captura solicitud de permiso de localización

4.2.4 Lectura de los datos con *Sensors API*.

La API de sensores como se explica en el anexo C, nos permite leer datos de los sensores de nuestro dispositivo en tiempo real, y esto es justo lo que necesitamos para mostrar durante el entrenamiento al usuario su velocidad, la distancia que ha recorrido, su frecuencia cardíaca…. Para ello se ha realizado una clase que se va a encargar de leer los datos de los sensores en *Google Fit*, a través de la API de sensores. Vamos a crear tantos objetos de esta clase como datos queramos obtener.

Él constructor de esta clase es el siguiente:

```
public ControlerSensor(Activity mActivity, DataType requestedDatatype, int requestedDataSource, int id, int type
    this.mActivity = mActivity;
    this.mRequestedDatatype = requestedDatatype;
    this.mRequestedDataSource = requestedDataSource;
    this.id = id;
    this.type = type;
```

Ilustración 4-14. Código constructor *ControlerSensor*

Los datos importantes para leer los datos en Google Fit son:

- *requestedDatatype*: Es el dato que queremos leer de los sensores de nuestro dispositivo en *Google Fit*.
- *requestedDataSource*: Es el tipo del dato, *TYPE_RAW* o *TYPE_DERIVED*

A continuación, vamos a ver los diferentes métodos que tiene esta clase para acceder a los datos que queremos.

En primer lugar, lo que vamos a hacer es comprobar si tenemos los permisos necesarios para poder leer el dato que queremos, los cuales deberíamos de tener si el usuario ya le ha dado el visto bueno cuando ha iniciado sesión con su perfil de Google, como se ha explicado anteriormente [13]:

```
public void createFitnessFile() {
    FitnessOptions fitnessOptions = FitnessOptions.builder()
            .addDataType(mRequestedDatatype)
            .build();

    if (!GoogleSignIn.hasPermissions(GoogleSignIn.getLastSignedInAccount(mActivity), fitnessOptions)) {
        GoogleSignIn.requestPermissions(
                mActivity, // your activity
                GOOGLE_FIT_PERMISSIONS_REQUEST_CODE,
                GoogleSignIn.getLastSignedInAccount(mActivity),
                fitnessOptions);
    } else {
        findFitnessDataSources();
    }

}
```

Ilustración 4-15. Código comprueba permisos en *Google Fit*

Una vez comprobado si el usuario ha concedido los permisos, lo que vamos a hacer es comprobar si el dato que vamos a solicitar es posible obtenerlo con nuestro dispositivo, para ello se ha implementado el siguiente método [13]:

```
public void findFitnessDataSources(){
    System.out.println("si");
    Fitness.getSensorsClient(mActivity, GoogleSignIn.getLastSignedInAccount(mActivity))
            .findDataSources(
                    new DataSourcesRequest.Builder()
                            .setDataTypes(mRequestedDatatype)
                            .setDataSourceTypes(mRequestedDataSource)
                            .build())
            .addOnSuccessListener(
                    new OnSuccessListener<List<DataSource>>() {
                        @Override
                        public void onSuccess(List<DataSource> dataSources) {
                            for (DataSource dataSource : dataSources) {
                                System.out.println("Data Source type: " + dataSource.getDataType().getName());

                                // Let's register a listener to receive Activity data!
                                if (dataSource.getDataType().equals(mRequestedDatatype)
                                        && mListener == null) {
                                    System.out.println("Data source found!  Registering.");
                                    accessGoogleFit(dataSource, mRequestedDatatype);
                                }
                            }
                        }
                    })
            .addOnFailureListener(
                    (e) → { System.out.println("failed"); });
}
```

Ilustración 4-16. Código busca el tipo de dato

En caso de que el dato sea posible obtenerlo con nuestro dispositivo, se llama al siguiente método que crea un *listener* en el cual vamos a ir obteniendo los valores del dato que hemos solicitado. [13]

```
public void accessGoogleFit(DataSource dataSource, DataType dataType) {
    mListener = new OnDataPointListener() {
        @Override
        public void onDataPoint(DataPoint dataPoint) {
            for (Field field : dataPoint.getDataType().getFields()) {
                Value val = dataPoint.getValue(field);
                double dato = formatearDecimales(Double.parseDouble(val.toString()),  numeroDecimales: 2);
                datosTraining.add(dato);
                if(id!=0)
                    addNewDato(Double.parseDouble(val.toString()));
                if(id == -1)
                    addFreqDato(Double.parseDouble(val.toString()));
                System.out.println("Detected DataPoint field: " + field.getName());
                System.out.println("Detected DataPoint value: " + val);
            }
        }
    };


    Fitness.getSensorsClient(mActivity, GoogleSignIn.getLastSignedInAccount(mActivity))
            .add(
                    new SensorRequest.Builder()
                            .setDataSource(dataSource) // Optional but recommended for custom data sets.
                            .setDataType(dataType) // Can't be omitted.
                            .setSamplingRate( l: 3, TimeUnit.SECONDS)
                            .build(),
                    mListener)
            .addOnCompleteListener(
                    new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if (task.isSuccessful()) {
                                System.out.println("Liostener register: ");
                            } else {
                                System.out.println("Liostener no register: " + task.getException());
                            }
                        }
```

Ilustración 4-17. Código obtiene los valores de los datos solicitados

4.2.5 Controlador alertas

Hay dos controladores muy similares importantes en los entrenamientos. Uno que se crea tanto en los entrenamientos libres como por intervalos, que es el *controllerAlertas* y el otro que se crea en caso de ser un entrenamiento por intervalos, que es el *controllerIntervalos*. Como ambos son muy similares se ha decidido explicar la clase de uno de ellos, la clase *controlerAlertas*.

Esta clase tiene como atributos:

- *Alerta*: Es un objeto alerta, la cual se va a controlar.
- *timeCountInMilliSeconds* : Es un tipo long que representa cada cuanto tiempo se va a notificar la alerta al usuario en caso de que esta sea de tipo *TIEMPO*.
- *restTime*: Es un tipo long que representa el tiempo que le queda a la alerta para ser notificada en caso de que el usuario pare el entrenamiento.
- *countDownTimer*: Es un objeto *CountDownTimer* que permite realizar una cuenta atrás en caso de que la alerta sea de tipo *TIEMPO*.
- *context*: Es el *Context* de la actividad del entrenamiento en la que se crea el controlador, que se necesitará para notificar la alerta.
- *Cancel*: Es un boolean que representa si una alerta ha sido cancelada o está habilitada.
- *timerStatus*: es un enum que representa si se ha parado la cuenta atrás o está en curso.

```
public class ControlerAlertas {
    private Alerta alerta;
    private long timeCountInMilliSeconds;
    private long restTime;
    private CountDownTimer countDownTimer;
    private Context context;
    private boolean cancel;

    private TimerStatus timerStatus = TimerStatus.STARTED;

    private enum TimerStatus {
        STARTED,
        STOPPED
    }

    public ControlerAlertas(Alerta alerta, Context context){
        this.alerta = alerta;
        this.context = context;
        if(alerta.isTime()){
            setTimerValues();
            startCountDownTimer();
        }
        this.cancel = false;
    }
```

Ilustración 4-18. Constructor y atributos de *ControlerAlertas*

Y tiene los siguientes métodos :

- *setTimerValues()*: registra el tiempo que se va a contar, con los minutos y segundos de la alarma.
- *startStop()*: para y reanuda el contador.
- *Reset()*: resetea la cuenta atrás.

```
/**
 * se inicializa el contador con los min y seg asignados por el usuario
 */
private void setTimerValues() {
    timeCountInMilliSeconds = alerta.getMin() * 60 * 1000 + alerta.getSeg() * 1000;
}

/**
 * para y reanuda el contador.
 */
public void startStop() {
    if (timerStatus == TimerStatus.STOPPED) {
        if(restTime != 0)
            timeCountInMilliSeconds = restTime;
        // changing the timer status to started
        timerStatus = TimerStatus.STARTED;
        // llamamos a reanudar el contador
        startCountDownTimer();

    } else {
        // llamamos a parar el contador
        timerStatus = TimerStatus.STOPPED;
        stopCountDownTimer();

    }

}

/**
 * resetea el contador
 */
public void reset() {
    setTimerValues();
    stopCountDownTimer();
    startCountDownTimer();
}
```

Ilustración 4-19. Código métodos *ControlerAlertas*

- *isFinishAlerta(double dato)*: comprueba si una alerta de tipo *DISTANCIA* o *CALORIAS* ha finalizado.

```
/*
* comprueba si una alerta de tipo distancia o calorias ha finalizado
*/
public boolean isFinishAlerta(double dato){
    return this.alerta.getDuracion() <= dato;
}
```

Ilustración 4-20. Código método *ControlerAlertas*

```
/**
 * comienza el contador
 */
public void startCountDownTimer() {
    countDownTimer = new CountDownTimer(timeCountInMilliSeconds,  countDownInterval: 1000) {
        @Override
        public void onTick(long millisUntilFinished) { restTime = millisUntilFinished; }

        @Override
        public void onFinish() {
            //creo la notificacion y reseteo la alarma
            NotificationBuilder.update(context, alerta);
            reset();
        }

    }.start();
    countDownTimer.start();
}


/**
 * para el contador
 */
public void stopCountDownTimer() { countDownTimer.cancel(); }


public boolean getCancel() { return cancel; }

/*
* si el usuario cancela la alarma o la habilita
*/
public void setCancel(boolean cancel) { this.cancel = cancel; }
```

Ilustración 4-21. Código métodos *ControlerAlertas*

- *startCountDownTimer()*: activa la cuenta atrás. En su interior cuenta con dos métodos que se requieren implementar al llamar al crear el objeto *CountDownTimer*:
    - *onTick(long milliUntilFinished)* : a este método se accede cada segundo que se cuenta, y en el guardamos el tiempo que resta para que se notifique la alarma.
    - *onFinish()*: este método se accede cuando finaliza la cuenta atrás, y en él se crea la notificación de la alerta y se resetea la alerta.
- *stopCountDownTimer()*: para la cuenta atrás.
- *getCancel() y setCancel*: son el getter y el setter de la variable cancel, para comprobar si la alerta ha sido cancelada y para cancelarla.

4.2.6 Notificaciones

Se ha utilizado las notificaciones de Android para comunicar al usuario durante el entrenamiento las alarmas que haya configurado. Estas notificaciones se crean en la clase *NotificationBuilder*, en la que se encuentra el siguiente método al cual llamaremos cuando queramos notificar algo al usuario:

```
public static void update(Context context, String descripcion, String titulo) {
    NotificationBuilder builder = new NotificationBuilder(context, descripcion, titulo);
    builder.buildNotification();
}
```

Ilustración 4-22. Código método *update()* de *NotificationBuilder*

Este método llama al constructor de la notificación, al cual se le pasan los siguientes parámetros:

- *Context*: el contexto de la actividad desde la que se llama al método.
- *Descripción*: la descripción que se quiere mostrar en la notificación.
- *Titulo*: el título de la notificación.

Y finalmente se llama al método *builNotification()*, que se encargará de crear y mostrar la notificación como vamos a ver en el siguiente código:

```
public void buildNotification() {

    int importance = NotificationManager.IMPORTANCE_DEFAULT;
    NotificationChannel channel = new NotificationChannel(NOTIFICATION_CHANNEL_ID,  name: "My Channel", importance);
    channel.setDescription("notificacion");
    channel.enableLights(true);

    //para que vibre al notificar
    channel.setVibrationPattern(new long[]{100, 200, 300, 400, 500, 400, 300, 200, 400});
    channel.enableVibration(true);

    NotificationManager mNotificationManager =
            (NotificationManager) mContext.getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.createNotificationChannel(channel);
    //Intent intent = new Intent(mContext, WorkoutViewActivity.class);
    //PendingIntent pendingIntent = PendingIntent.getActivity(mContext, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);

    NotificationCompat.Builder builder = new NotificationCompat.Builder(mContext, NOTIFICATION_CHANNEL_ID);
    builder.setContentTitle(titulo);
    builder.setContentText(descripcion);
    builder.setSmallIcon(R.drawable.alertas);
    builder.setDefaults(Notification.DEFAULT_ALL);
    //builder.setContentIntent(pendingIntent);

    NotificationCompat.WearableExtender wearableExtender =
            new NotificationCompat.WearableExtender()
                    .setCustomSizePreset(Notification.WearableExtender.SIZE_DEFAULT);
                    //.setDisplayIntent(pendingIntent);

    builder.extend(wearableExtender);
    Notification notification = builder.build();

    NotificationManagerCompat managerCompat = NotificationManagerCompat.from(mContext);
    managerCompat.notify(NOTIFICATION_ID, notification);
}
```

Ilustración 4-23. Código método *buildNotification()* de *NotificationBuilder*

5 Pruebas y resultados

Para comprobar el correcto funcionamiento de la aplicación se han realizado las siguientes pruebas, intentando englobar la mayoría de las funcionalidades. Estas pruebas se han realizado mediante el smartwatch *Huaweii Watch 2* que se me ha proporcionado.

- **El usuario inicia por primera vez la aplicación**: En primer lugar, se le muestra el logo de la aplicación y posteriormente inicia sesión con *Google* y da permisos a la aplicación para interactuar con *Google Fit*. Posteriormente indica los datos que le pide la aplicación y acepta los permisos para acceder a los sensores de las constantes vitales del reloj. Se le mide la frecuencia cardiaca. A partir de los datos registrados la aplicación calcula las zonas de frecuencia cardiaca del usuario como se dice en el anexo D y es registrado correctamente.

- **El usuario crea y configura un entrenamiento libre**: El usuario crea un entrenamiento libre con una intensidad libre. Después configura los campos que quiere que se muestren en la pantalla escogiendo la velocidad, el tiempo y la distancia, y añade una alerta para que le avise cada minuto de entrenamiento. Finalmente, el usuario empieza el entrenamiento, y se observan los campos que él había elegido, y como había configurado cada minuto el reloj vibra y le muestra una notificación en la que se le comunica que lleva un minuto de entrenamiento.

- **El usuario finaliza el entrenamiento y ve las estadísticas:** El usuario decide finalizar el entrenamiento, y la aplicación le muestra las estadísticas que la aplicación ha ido registrando al monitorizar su actividad física.

- **El usuario consulta su historial:** Después de haber realizado el entrenamiento, comprueba si el entrenamiento ha sido añadido a su historial. Efectivamente se muestra el entrenamiento añadido correctamente.

- **El usuario crea un entrenamiento por intervalos:** El usuario crea un entrenamiento por intervalos. Configura el entrenamiento con 3 repeticiones, de 3 minutos cada una y con un descanso entre ellas de 20 segundos. Comienza el entrenamiento, y la aplicación comunica automáticamente al pasar los primeros 3 minutos mediante una notificación que hace que vibre el smartwatch y comunica que el primer intervalo ha finalizado. Comienza los 20 segundos de descanso, y al acabar comienza el siguiente intervalo. Una vez han terminado todos los intervalos la aplicación le muestra al usuario las estadísticas de cada intervalo (También ha sido realizado satisfactoriamente con intervalos de distancia).

Las anteriores pruebas han tenido un resultado satisfactorio, por lo que podemos decir que los objetivos de este proyecto han sido cumplidos con éxito.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Al finalizar el trabajo de fin de grado, se ha conseguido realizar una aplicación para smartwatchs de entrenamiento. El usuario puede realizar tanto entrenamientos normales como por intervalos y configurar los entrenamientos a su gusto.

Podemos decir que el resultado es bueno, ya que se ha conseguido cumplir con los objetivos que se habían enunciado previamente. En la realización del proyecto nos hemos encontrado con dos dificultades por encima del resto:

- Realizar una aplicación para smartwatch con Android Wear. Tener muy en cuenta el tamaño con el que se cuenta para realizar un buen diseño de la interfaz gráfica y familiarizarse con Android Wear.
- Familiarizarse con la API de *Google Fit*, para realizar correctamente las funcionalidades en las que había que interactuar con ella.

También se ha puesto en práctica muchos de los conocimientos adquiridos durante la carrera en el proyecto. Principalmente han sido de mucha utilidad la asignatura de *Aplicaciones móviles*, a la hora de programar en Android, y las asignaturas de proyecto de *análisis y diseño de software* e *ingeniería del software*, a la hora de realizar el análisis y el diseño del proyecto.

6.2 Trabajo futuro

El trabajo realizado ha cumplido con los objetivos fundamentales previstos, pero la aplicación podría mejorar en algunos aspectos en el futuro, como pueden ser:

- Añadir a la aplicación aparte del entrenamiento de los tres deportes por separado, una modalidad de triatlón que unifique los 3 deportes en un mismo entrenamiento.
- Añadir un mapa para que el usuario pueda ver la ruta que está realizando, y si le gusta que la pueda guardar (Maps API).
- Añadir una versión en inglés de la aplicación.
- Realizar un backup de los datos del usuario y de los entrenamientos que haya realizado, incluyendo sus estadísticas.
- Que el usuario pueda repetir un entrenamiento que haya realizado anteriormente, y se cargue con su configuración completa.

Referencias

[1] PreMarathon. (26 de febrero de 2018). *youtube*. Obtenido de programar entrenamientos por intervalos Garmin: https://www.youtube.com/watch?v=G8Nd6dfwZyk&t=184s

[2] Palabra de Runner. (16 de Enero de 2017). *Youtube*. Obtenido de Programar entrenamientos por intervalos polar: https://www.youtube.com/watch?v=LQilPsFpits&t=535s

[3] Strava. (s.f.). *Google Play*. Obtenido de Strava: https://play.google.com/store/apps/details?id=com.strava

[4] Runtastic. (s.f.). *Google Play*. Obtenido de Runtastic: https://play.google.com/store/apps/details?id=com.runtastic.android

[5] RunKeeper. (s.f.). *Google Play*. Obtenido de RunKeeper: https://play.google.com/store/apps/details?id=com.fitnesskeeper.runkeeper.pro

[6] Nike. (s.f.). *Google Play*. Obtenido de Nike: https://play.google.com/store/apps/details?id=com.nike.plusgps

[7] Google. (s.f.). *Google Play*. Obtenido de Google fit: https://play.google.com/store/apps/details?id=com.google.android.apps.fitness

[8] González, J. (s.f.). *Respuestas plan difusión*. Obtenido de respuestas: https://docs.google.com/spreadsheets/d/1cPbOMvn0mJE6mwAPbIABCnOi3A1E6h-9Nvp8dZdOH-s/edit#gid=627881696

[9] González, J. (s.f.). *Respuestas plan difusión*. Obtenido de Respuestas: https://docs.google.com/spreadsheets/d/1RF-wIDPomWGy0cYwjD40tALkhGHU4YWuEmcUyLgVqiA/edit#gid=1

[10] Junta de Andalucía. (s.f.). *Marco de desarrollo de la junta de Andalucía*. Obtenido de Guía para la redacción de casos de uso: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416

[11] Google Developers. (17 de Abril de 2018). *Developer.android*. Obtenido de Standalone wear apps: https://developer.android.com/training/wearables/apps/standalone-apps

[12] Google Developers. (21 de Mayo de 2018). *Google sign-in for android*. Obtenido de Add Google Sign-In to Your Android App: https://developers.google.com/identity/sign-in/android/

[13] Google Developers. (14 de Diciembre de 2017). *Google fit*. Obtenido de Access raw sensor data: https://developers.google.com/fit/android/sensors

[14] Huawei Technologues. (s.f.). *Huawei*. Obtenido de Huawei Watch 2: https://consumer.huawei.com/es/wearables/watch2/specs/

[15] Apple. (s.f.). *Apple*. Obtenido de Apple Watch Series 3: https://www.apple.com/es/apple-watch-series-3/#sports-watch

[16] Fitbit. (s.f.). *Fitbit*. Obtenido de Fitbit Versa: https://www.fitbit.com/es/shop/versa

[17] Garmin. (s.f.). *Garmin*. Obtenido de Forerunner 645: https://buy.garmin.com/es-ES/ES/p/591311/pn/010-01863-10#specs

[18] Samsung. (s.f.). *Samsung*. Obtenido de Samsung gear fit pro 2: https://www.samsung.com/us/explore/gear-fit2-pro/

[19] Xiaomi. (s.f.). *Xiaomi*. Obtenido de Amazfit PACE: http://www.mi.com/es/amazfit-pace/specs/

[20] SUUNTO. (s.f.). *SUUNTO*. Obtenido de SUUNTO 3 fitness: http://www.suunto.com/es-ES/Productos/Relojes-deportivos/suunto-3-fitness/suunto-3-fitness-all-black/

[21] Polar España. (s.f.). *Polar*. Obtenido de Polar M600: https://www.polar.com/es/productos/sport/M600-GPS-smartwatch

[22] Google Developers. (20 de Septiembre de 2017). *Google fit*. Obtenido de Platform overview: https://developers.google.com/fit/overview

[23] Google Developers. (20 de Septiembre de 2017). GoogleFit. Obtenido de Get an OAuth 2.0 Client ID: https://developers.google.com/fit/android/get-api-key

[24] Google Developers. (14 de Diciembre de 2017). *Google Fit*. Obtenido de Android APIs: https://developers.google.com/fit/android/

[25] Google Developers. (11 de Enero de 2018). *Google fit*. Obtenido de Authorization on Android: https://developers.google.com/fit/android/authorization

[26] Google Developers. (12 de Diciembre de 2017). *Google fit*. Obtenido de Fitness data types: https://developers.google.com/fit/android/data-types

[27] Moya, P. (3 de Enero de 2016). *palabra de runner*. Obtenido de Zonas de pulsaciones: https://www.palabraderunner.com/karvonen-calcular-zonas-de-pulsaciones#formula-de-karvonen-para-calcular-tus-zonas-de-entrenamiento-por-pulsaciones

[28] Polar España. (2016 de Junio de 2016). *Polar*. Obtenido de Frecuencia cardíaca en reposo: https://www.polar.com/es/acerca_de_polar/noticias/c_mo_calcular_tu_frecuencia_card_aca_en_reposo

Glosario

API	Application Programming Interface
GPS	Global Position System, es un sistema de navegación basado en 24 satélites, en órbita sobre la tierra que envían información sobre la posición de una persona o cosa.
UML	Unified Modeling Language, es un estándar para crear esquemas, diagramas y documentación relativa a los desarrollos de software.
Framework	Arquitectura de software con una serie de métodos y herramientas, que sirven para el desarrollo de un proyecto.
Smartphone	Teléfono móvil con pantalla táctil, que permite al usuario conectarse a Internet, gestionar su correo electrónico y descargarse aplicaciones para diferentes utilidades.
Smartwatch	Reloj inteligente, que permite al usuario accede a Internet, recibir y realizar llamadas telefónicas, enviar y recibir mensajes y corres electrónicos, recibir notificaciones del smartphone…. Estas funcionalidades las tienen también los smartphone, pero estos dispositivos cada vez son más utilizados para hacer deporte por la comodidad que supone.

Anexos

A Relojes deportivos

MARCA	MODELO	MEMORIA	BATERÍA	PANTALLA	ÍNDICE DE RESISTENCIA AL AGUA Y AL POLVO	PRECIO (€)
HUAWEI	HUAWEI 2.0	4 GB Flash + 768 MB RAM*	420 mAh* 410 mAh* (valor mínimo)	Pantalla AMOLED circular de 1,2" 390 x 390 píxeles con una PPI de 326 Corning Gorilla Glass	IP68	379?
	Apple Watch Series 3	512MB	Batería de iones de litio recargable integrada Hasta 18 horas	Pantalla Retina OLED de segunda generación con Force Touch dos veces más brillante (1,000 nits) Vidrio Ion-X (reforzado) 272 x 340 píxeles (38 mm) 312 x 390 píxeles (42 mm)	5 ATM a 50 metros	Desde 369 €
fitbit	fitbit Versa	4 GB	Más de 4 días	Ancho Diagonal: 34,0 mm Tamaño de la pantalla 24,075 x 24,075 mm		199,95 €
GARMIN	Forerunner® 645	200 horas de datos de actividad	Hasta 7 días en modo reloj inteligente 5 horas en modo GPS	Visible a la luz del sol, transflectiva, memoria a nivel de píxeles (MIP) Tamaño de la pantalla Diámetro de 30,4 mm (1.2") Resolución 240 x 240 píxeles	Resistente al agua 5ATM	449,99 €

Ilustración 0-1. Relojes deportivos1

Ilustración 0-2. Relojes deportivos 2

Las referencias con las que se han obtenido la información de estas ilustraciones, por orden, son: [14], [15], [16], [17], **Error! Reference source not found.**, [18], [20] y [21]

B Cuestionario

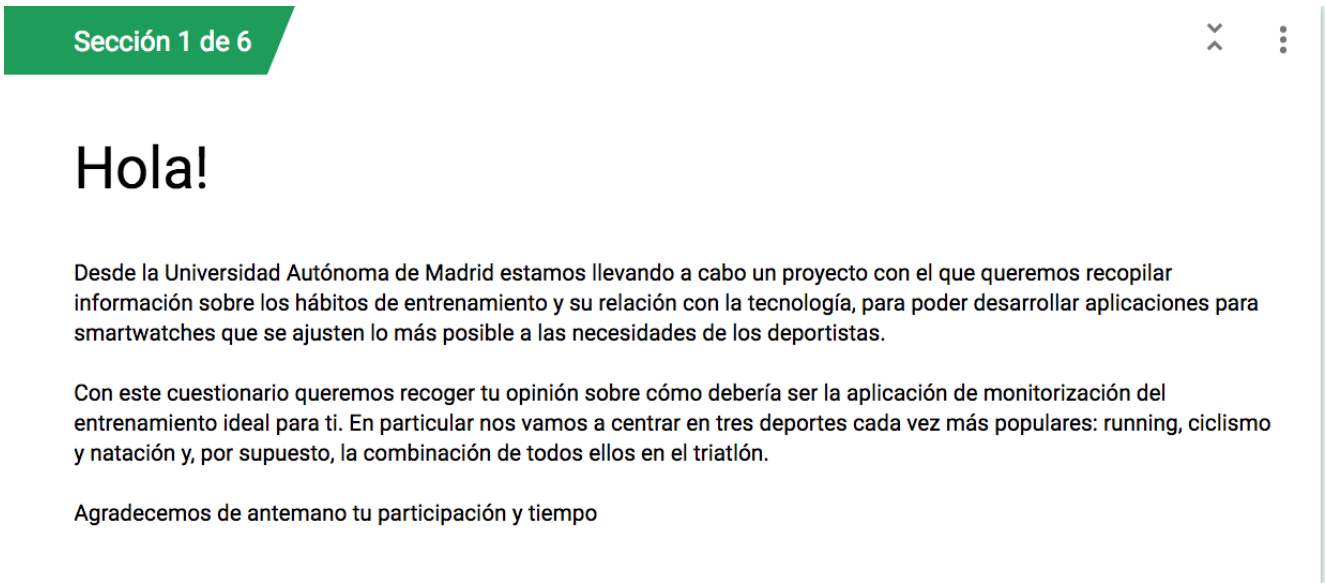

Ilustración 0-3. Introducción cuestionario

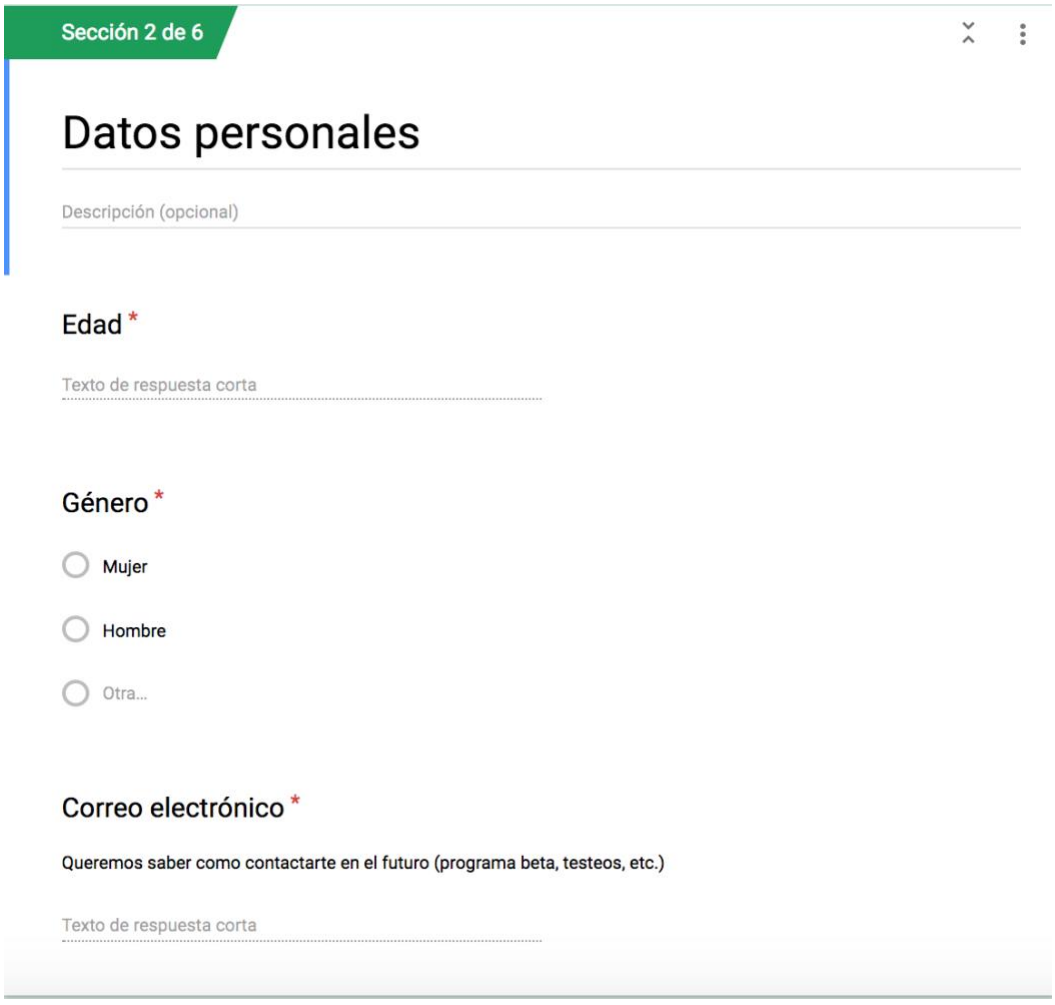

Ilustración 0-4. Datos personales cuestionario

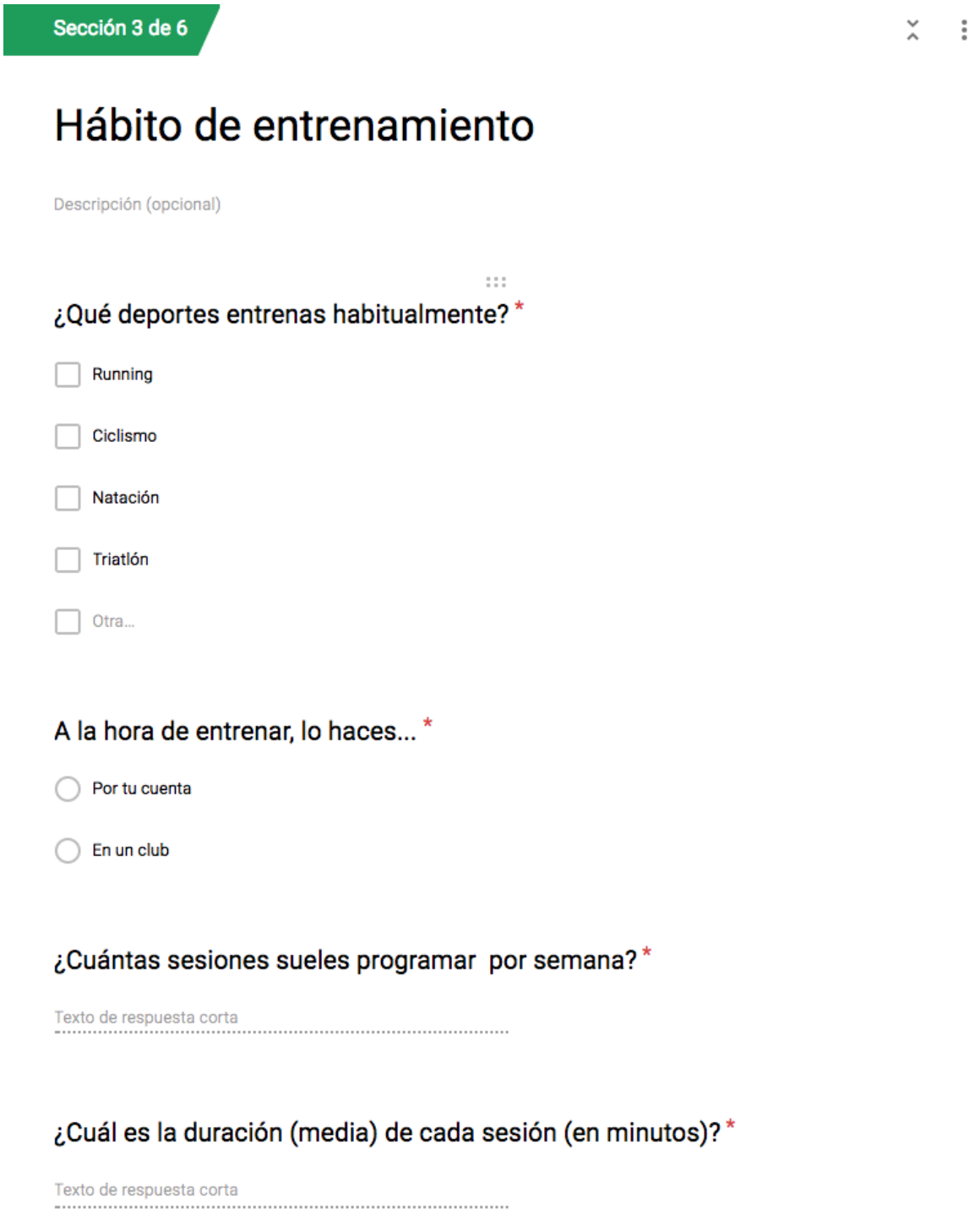

Ilustración 0-5. Hábitos entrenamiento cuestionario

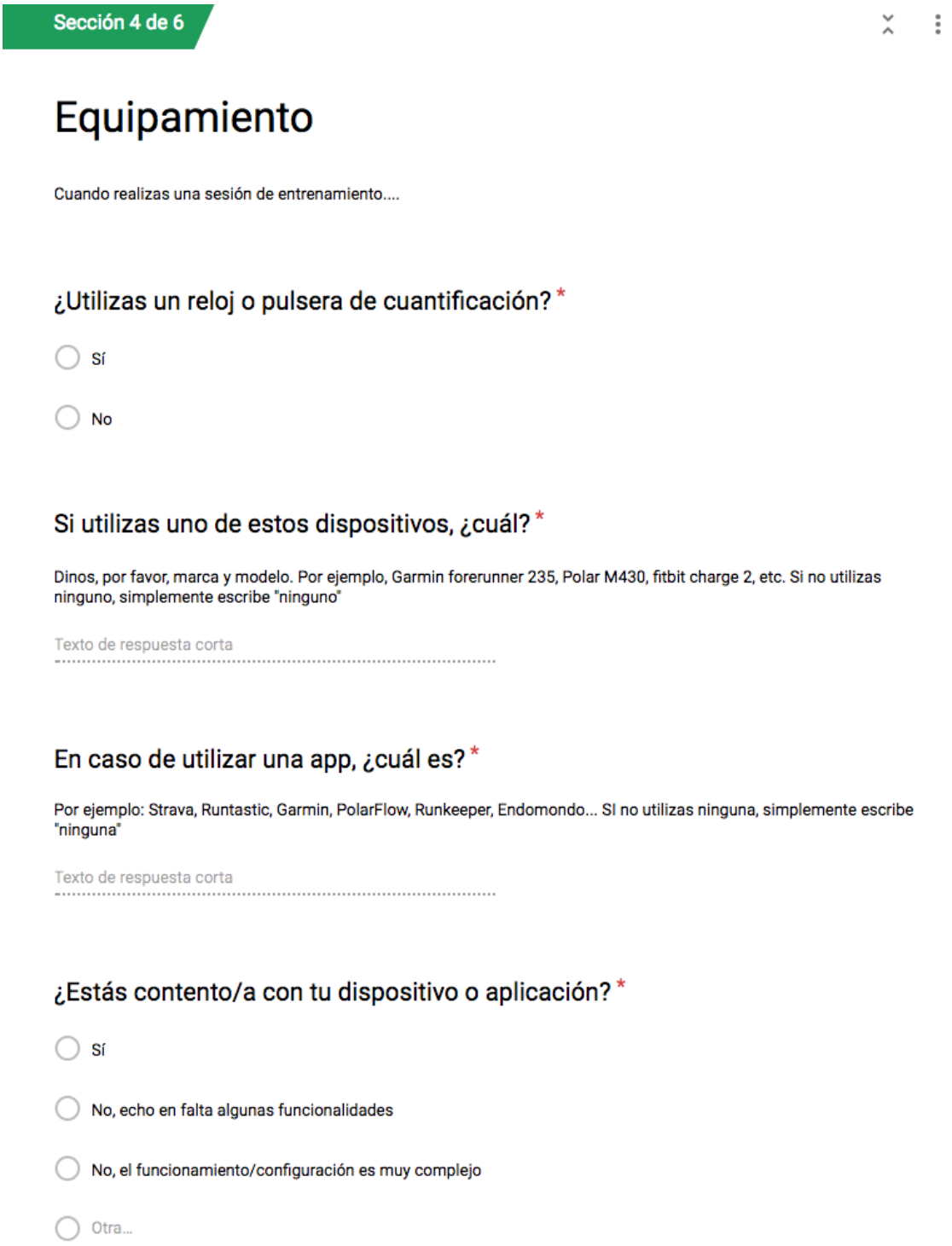

Ilustración 0-6. Equipamiento cuestionario

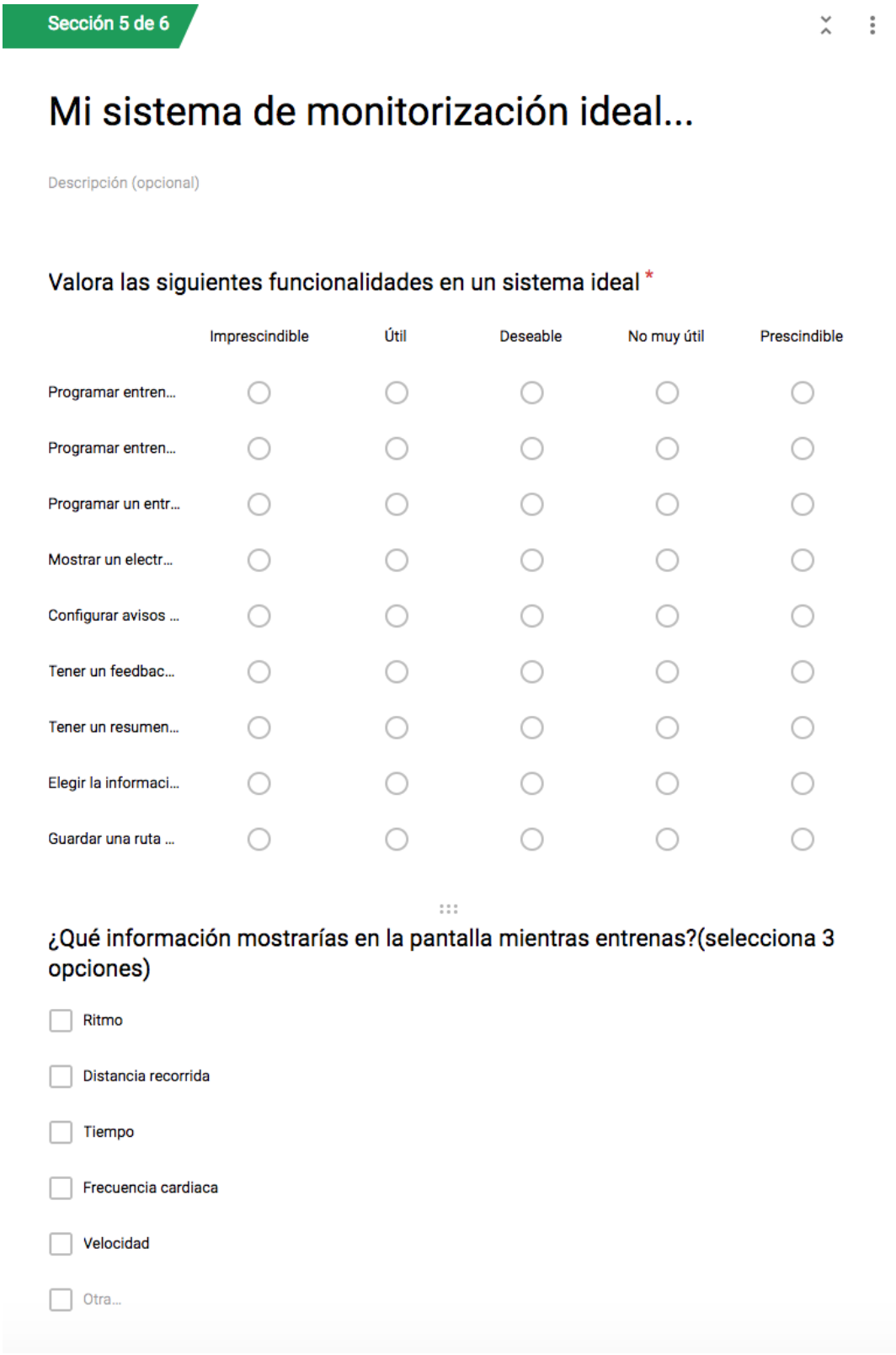

Ilustración 0-7. Sistema de monitorización cuestionario

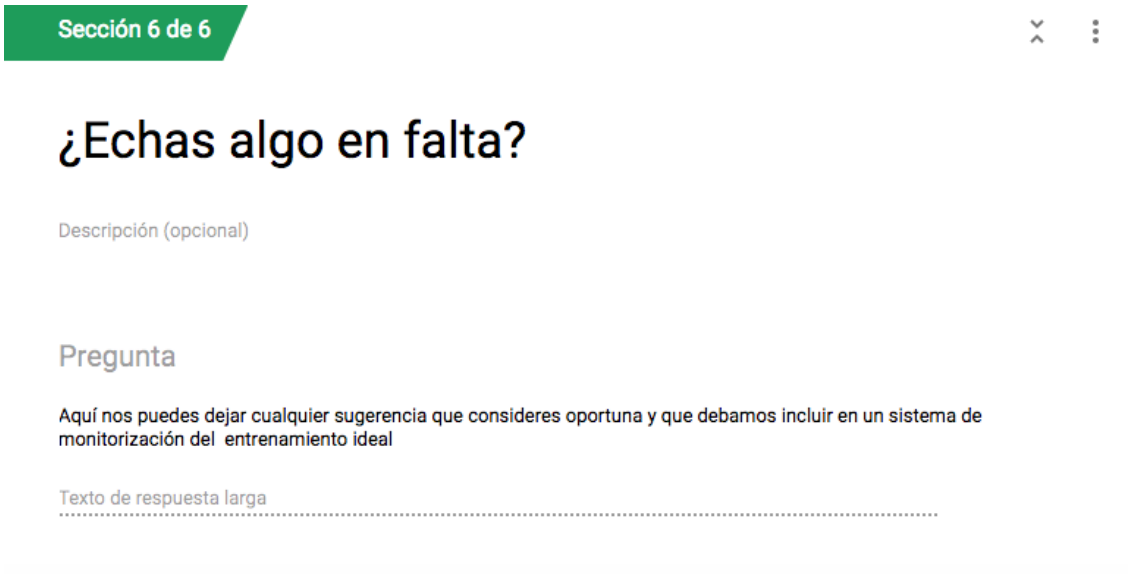

Ilustración 0-8. Echas algo en falta. Cuestionario

A continuación, se muestran una serie de gráficas que se mencionan en el documento, que muestran un análisis de algunas respuestas de los usuarios al cuestionario:

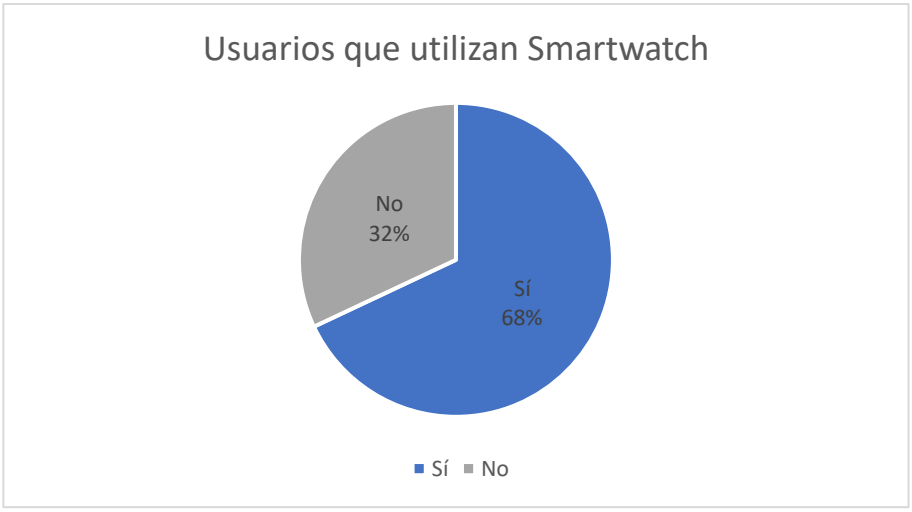

Ilustración 0-9. Gráfica usuarios que utilizan smartwatch

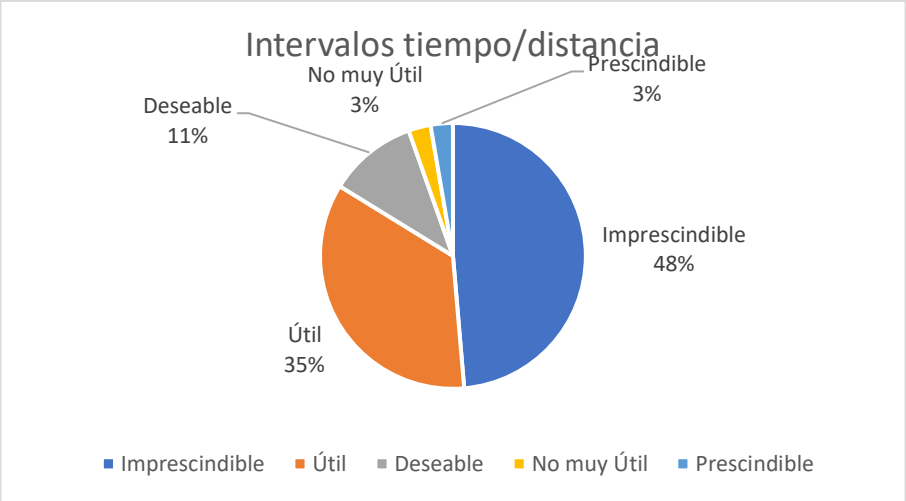

Ilustración 0-10. Gráfica respuestas intervalos tiempo/distancia.

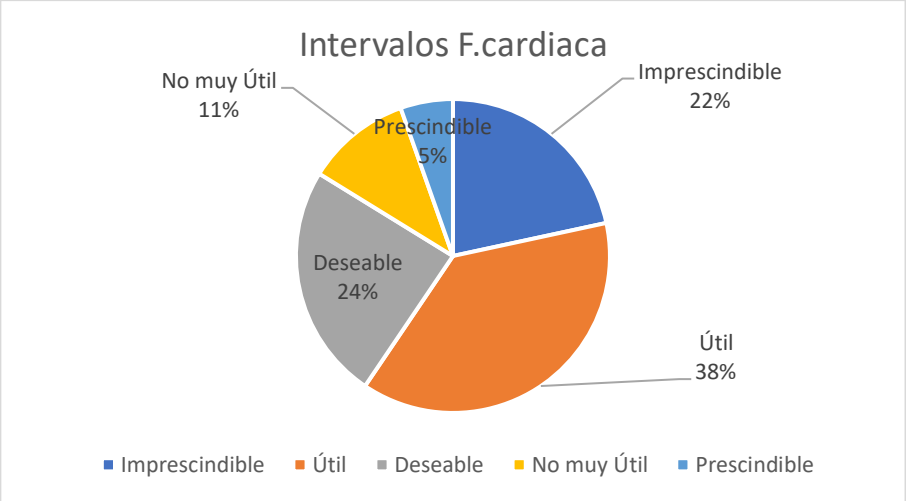

Ilustración 0-11. Gráfica respuestas intervalos frecuencia cardiaca.

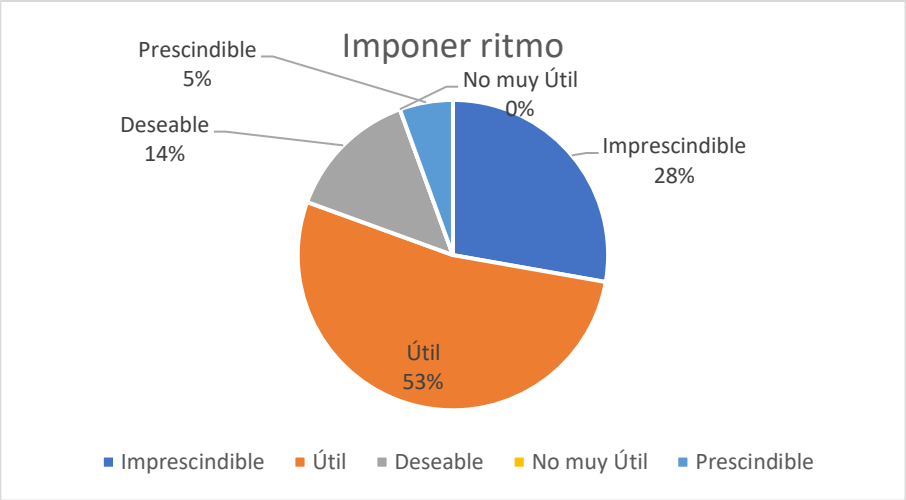

Ilustración 0-12. Gráfica respuestas imponer ritmo.

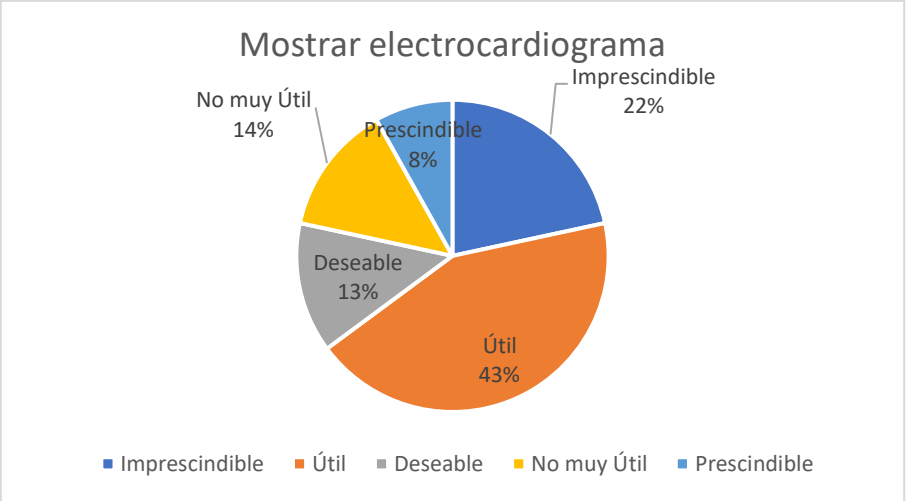

Ilustración 0-13. Gráfica respuestas mostrar electrocardiograma.

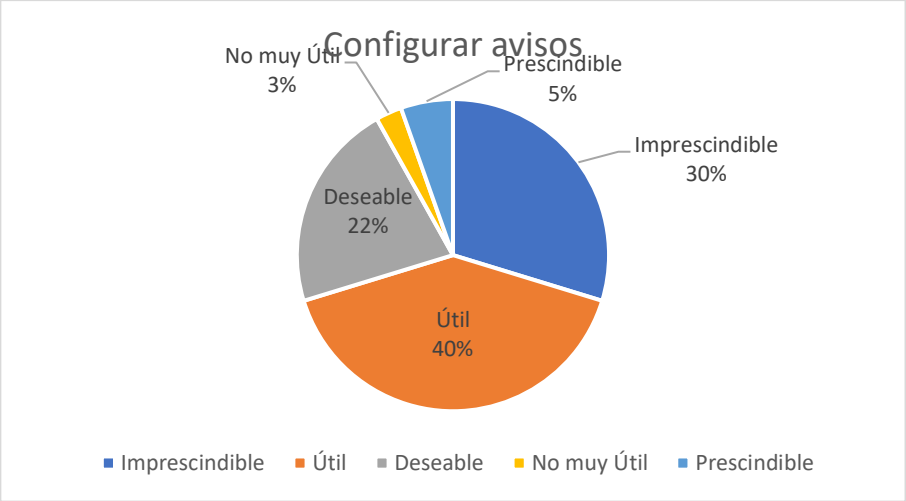

Ilustración 0-14. Gráfica respuestas configurar avisos.

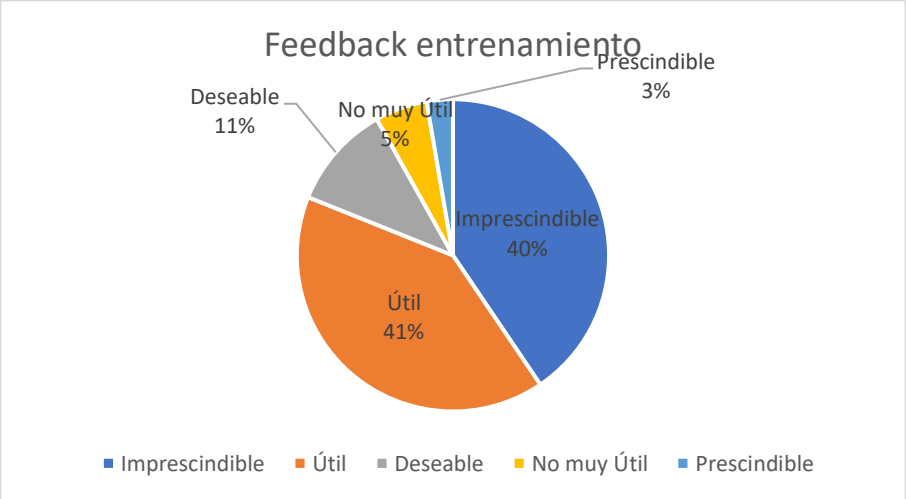

Ilustración 0-15. Gráfica respuestas feedback entrenamiento.

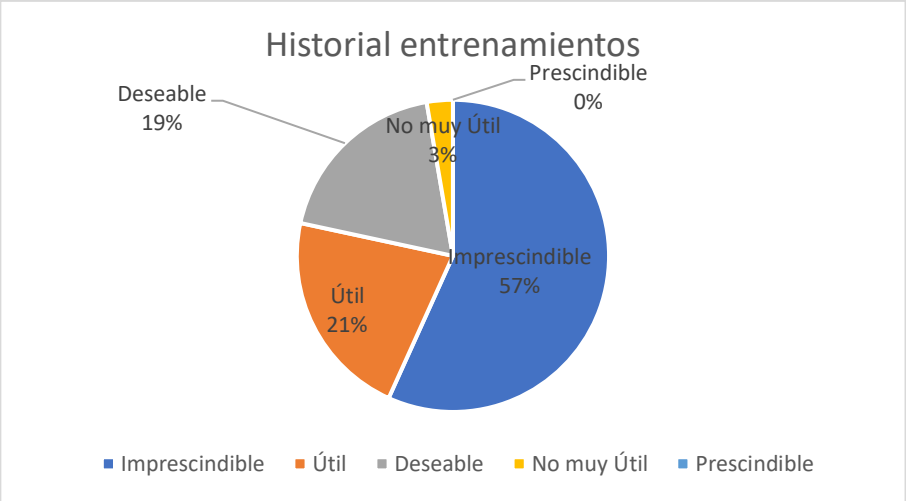

Ilustración 0-16. Gráfica respuestas historial entrenamientos.

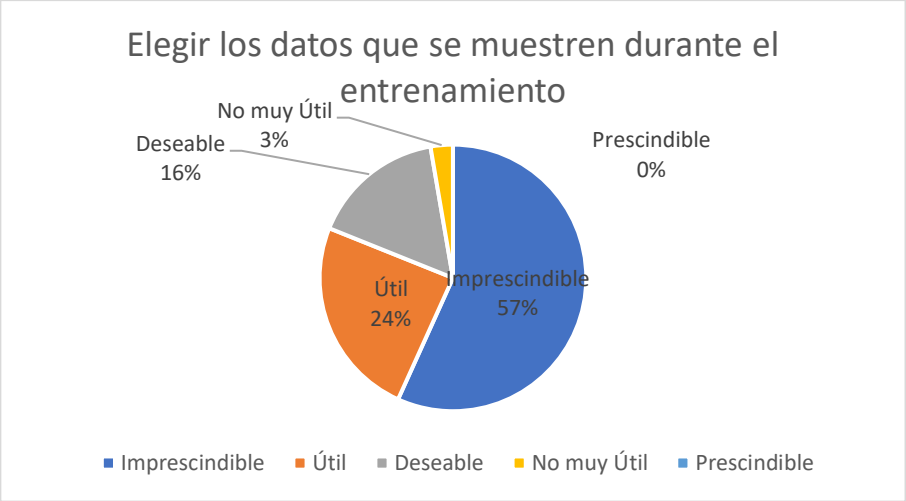

Ilustración 0-17. Gráfica respuestas elección de datos que se muestran durante el entrenamiento.

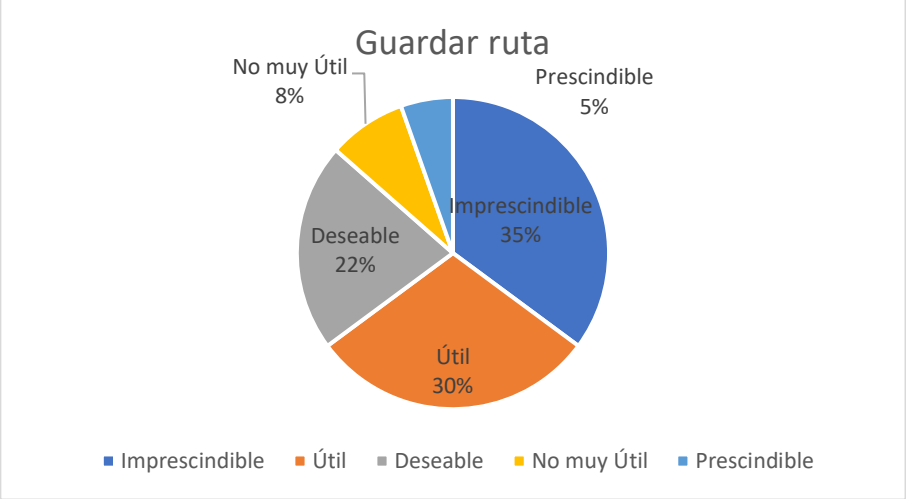

Ilustración 0-18. Gráfica respuestas guardar ruta.

C Google fit

Google Fit permite almacenar datos fitness en un repositorio central y acceder a ellos mediante APPs y dispositivos:

- Las aplicaciones deportivas pueden almacenar datos de cualquier sensor o dispositivo.
- Las aplicaciones deportivas pueden acceder a datos almacenados por cualquier otra aplicación.
- Los datos de los usuarios se mantienen a pesar de las actualizaciones.

La siguiente figura representa la conexión entre los sensores o dispositivos, las APPs y el repositorio central de *Google Fit* [22]:

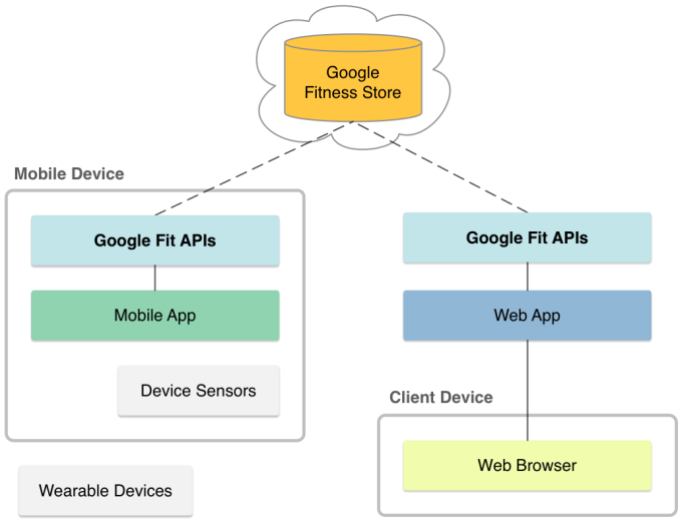

Ilustración 0-19. Conexiones *Google Fit*

Google Fit contiene los siguientes componentes:

- Repositorio central: Repositorio donde se almacenan los datos obtenidos por los sensores o dispositivos.
- Un *framework* de sensores: Conjunto de representaciones que nos facilitan interactuar con el repositorio.
- Permisos y control del usuario: Es una serie de permisos que se solicitan al usuario para poder trabajar y acceder con sus datos de fitness.
- Las APIs de Google Fit: Android y REST APIs que nos permiten acceder al repositorio y poder así realizar aplicaciones.

En este caso se utiliza la API de Android, para lo cual hemos tenido que realizar los pasos explicados en [23] para obtener una autorización y así conectar nuestra aplicación con la API para poder trabajar con ella correctamente.

La API nos ofrece los siguientes servicios [24]:

- **Sensors API**: Proporciona los datos obtenidos por los sensores de los dispositivos o de los dispositivos complementarios.
- **Recording API**: Proporciona el almacenamiento automático de datos fitness.
- **History API**: Permite el acceso al historial del usuario y borrar, insertar o leer los datos que tiene.
- **Sessions API**: Permite almacenar datos, con metadatos de sesiones.
- **Goals API**: Permite rastrear los objetivos que el usuario se ha establecido para su progreso físico y saludable.
- **Bluetooth Low Energy API**: Esta API permite a las aplicaciones buscar dispositivos disponibles y almacenar los datos que obtienen por bluetooth.
- **Config API**: Proporciona tipos de datos personalizados y configuraciones en Google Fit.

Antes de nada, para poder interactuar con los diferentes clientes que nos proporciona la API de Google Fit, debemos garantizar el acceso a los datos de los clientes. Estos permisos y los niveles de acceso lo representan los *scopes*. Cada *scope* proporciona acceso a un conjunto de datos diferentes y algunos proporcionan acceso de lectura únicamente y otros accesos de lectura y escritura. En la siguiente ilustración podemos ver los *scopes* [25]:

Permission	Scope	Type of Access	Data Types
Activity	FITNESS_ACTIVITY_READ	Read	com.google.activity.sample com.google.activity.segment com.google.activity.summary (deprecated) com.google.calories.consumed com.google.calories.expended com.google.cycling.pedaling.cadence com.google.power.sample com.google.step_count.cadence com.google.step_count.delta com.google.activity.exercise
	FITNESS_ACTIVITY_READ_WRITE	Read and Write	
Body	FITNESS_BODY_READ	Read	com.google.heart_rate.bpm com.google.heart_rate.summary com.google.height com.google.weight com.google.weight.summary
	FITNESS_BODY_READ_WRITE	Read and Write	
Location	FITNESS_LOCATION_READ	Read	com.google.cycling.wheel_revolution.cumulative com.google.cycling.wheel.revolutions com.google.distance.delta com.google.location.sample com.google.location.bounding_box com.google.speed com.google.speed.summary
	FITNESS_LOCATION_READ_WRITE	Read and Write	
Nutrition	FITNESS_NUTRITION_READ	Read	com.google.nutrition.item com.google.nutrition.summary
	FITNESS_NUTRITION_READ_WRITE	Read and Write	

Ilustración 0-20. Scopes *Google Fit*

Por otro lado, hay que conocer los tipos de datos que Google Fit proporciona, para solicitarlos para leerlos o escribirlos. En las siguientes ilustraciones se muestran tipos de datos de Google Fit [26]:

Data Type Name	Description	Permission	Fields (Format—Unit)
com.google.activity.sample	Instantaneous sample of the current activity.	Activity	activity (**int**—enum) confidence (**float**—percent)
com.google.activity.segment	Continuous time interval of a single activity.	Activity	activity (**int**—enum)
(deprecated) **com.google.calories.consumed**	Total calories consumed over a time interval.	Activity	calories (**float**—kcal)
com.google.calories.expended	Total calories expended over a time interval.	Activity	calories (**float**—kcal)
com.google.cycling.pedaling.cadence	Instantaneous pedaling rate in crank revolutions per minute.	Activity	rpm (**float**—rpm)
com.google.cycling.wheel_revolution.rpm	Instantaneous wheel speed.	Location	rpm (**float**—rpm)
com.google.distance.delta	Distance covered since the last reading.	Location	distance (**float**—meters)
com.google.heart_rate.bpm	Heart rate in beats per minute.	Body	bpm (**float**—bpm)
com.google.height	The user's height, in meters.	Body	height (**float**—meters)
com.google.location.sample	The user's current location.	Location	latitude (**float**—degrees) longitude (**float**—degrees) accuracy (**float**—meters) altitude (**float**—meters)

Ilustración 0-21. Datos *Google Fit*

com.google.nutrition	Food item information	Nutrition	nutrients (**Map<String, float>**—calories/grams/IU) meal_type (**int**—enum) food_item (**String**—n/a)
com.google.power.sample	Instantaneous power generated while performing an activity.	Activity	watts (**float**—watts)
com.google.speed	Instantaneous speed over ground.	Location	speed (**float**—m/s)
com.google.step_count.cadence	Instantaneous cadence in steps per minute.	Activity	rpm (**float**—steps/min)
com.google.step_count.delta	Number of new steps since the last reading.	Activity	steps (**int**—count)
com.google.weight	The user's weight.	Body	weight (**float**—kg)
com.google.activity.exercise	A user's continuous workout routine.	Activity	exercise (**int**—enum) repetitions (**int**—count) resistance type (**int**—enum) resistance (**float**—kg) duration (**int**—milliseconds)

Ilustración 0-22. Datos *Google fit*

Finalmente, para interactuar con los datos te tienes que conectar a alguna de las API que hemos enumerado anteriormente, para lo cual antes era necesario crear un objeto *GoogleApiClient*, pero que ahora desde la versión de *Google play Services* 11.6.0, simplemente al hacer un *getter* del cliente que quieras utilizar (se puede ver un ejemplo en la ilustración x en la línea que pone *Fitness.getSensorsClient*…), este se encarga automáticamente de conectarse con *Google Play Services*.

D Cálculo frecuencia máxima y zonas cardiacas

Para realizar el cálculo de las zonas de frecuencia cardiaca o zonas de entrenamiento por pulsaciones vamos a utilizar la fórmula de Karvonen [27]. Esta fórmula tiene en cuenta la frecuencia cardiaca máxima y la frecuencia cardiaca en reposo para calcular un porcentaje de esfuerzo.

$$\%\ de\ FCobjetivo = ((FCmax - FCrep) \times \%intensidad) + FCrep$$

Lo primero que vamos a realizar es calcular la frecuencia cardiaca máxima como se dice en [27], utilizando la fórmula propuesta por Tanaka, Monahan y Seals:

$$FCmax = 208 - (0{,}7 \times edad)$$

Después se realiza el cálculo de la frecuencia cardiaca en reposo, como se explica en [28], lo cual lo hará nuestra aplicación como ya se ha explicado anteriormente, tomando medidas de las pulsaciones por minuto del usuario durante 2-3 minutos.

Finalmente, una vez que ya tenemos todos los datos necesarios para realizar el cálculo, simplemente sustituimos los datos en la fórmula de Karvonen.

Mediante esta fórmula se realiza el cálculo de las zonas de frecuencia cardiaca del usuario como se explica en [27]:

Zona1 50-60%: calentamiento, rehabilitación, enfriamiento….
Zona2 60-70%: ritmo cómodo.
Zona3 70-80%: ritmo moderado.
Zona4 80-90%: intensidad mayor para mejorar el rendimiento.
Zona5 90-100%: Máximo esfuerzo. Mayor intensidad que podemos soportar.